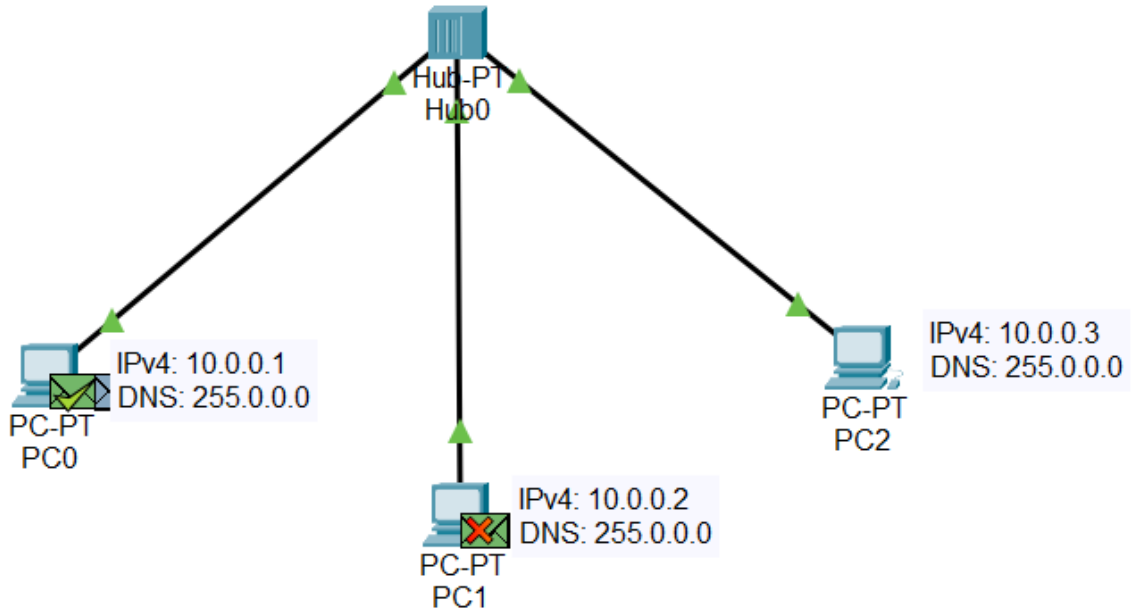
COMPUTER NETWORKS

LABORATORY PROGRAM – 1

Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC2	ICMP		0.000	N	0	(edit)	

```
C:\>ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3: bytes=32 time=9ms TTL=128
Reply from 10.0.0.3: bytes=32 time<1ms TTL=128
Reply from 10.0.0.3: bytes=32 time=1ms TTL=128
Reply from 10.0.0.3: bytes=32 time<1ms TTL=128

Ping statistics for 10.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 9ms, Average = 2ms
```

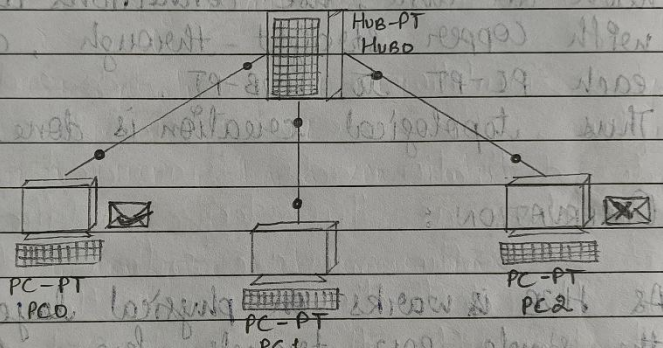
1.10.24 DATE: PAGE:

LABORATORY PROGRAM - 1

⇝ AIM OF THE EXPERIMENT : To demonstrate the transmission of a simple PDU between two devices connected using a hub and a switch.

⇝ TOPOLOGY :

- STAR TOPOLOGY [HUB] :



- CONNECTION : COPPER STRAIGHT-THROUGH

→ CONFIGURATION :

1. Refer topological procedure for the connection and now configuration of PC's takes place.
2. Click on PC-PT [PC0], go to config, then the Interface, FastEthernet0.
3. In IPv4, assign IP address as 10.0.0.1 and the subnet mask gets automatically defined.

DATE: PAGE:

4. Now, similarly config other generic PC-PT's. And click on simple PDU, from source-destination

⇝ TOPOLOGICAL CREATION :

1. In end devices, use generic PC-PT and place it on the workspace. Drag atleast 3 generic PC-PT.
2. Now, using ~~or~~ HUB, place generic HUB-PT on workspace.
3. With all done, use connections and with copper straight-through, connect each PC-PT to HUB-PT.
4. Thus topological creation is done.

⇝ OBSERVATION :

- As HUB is works on physical layer, the simple PDU travels from source [PC-PT PC0] to destination [PC-PT, PC1] from HUB-PT being as a intermediate.
- Now, when we click on simulation mode, then on Auto Capture / Play, the simulation of simple PDU is displayed.
- At first, the message traverse from source [PC0] to the hub0 and hub0 being non-intelligent intellectual, the message is transmitted to rest other devices.
- As we selected the destination [PC1], after message recieved, it sends backs the signal to hub0, then to rest

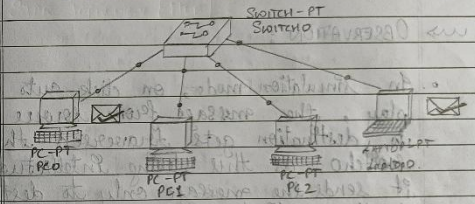
DATE: PAGE:

devices. Thus, the source [PC0] get the message that, destination or the aim is accomplished.

⇝ TRANSMISSION USING SWITCH

⇝ TOPOLOGY :

- STAR TOPOLOGY [SWITCH] :



- CONNECTION : Copper straight-through

⇝ TOPOLOGY CREATION :

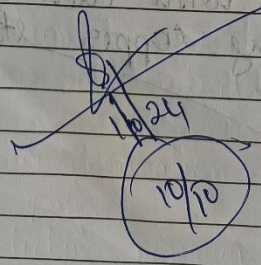
- Using End devices, drag and drop the generic PC-PT's and laptop-PT.
- Now, take generic switch, add to PT, and connect it with each generic end devices using copper straight through.

CONFIGURATION :

- On clicking each generic end device, config each devices of IPv4 as 10.0.0.1 and so on for each. The subnet mask get automatically generated.
- Now, with simple PDU; click on source [PC-PT PC0] to the destination [Laptop0].

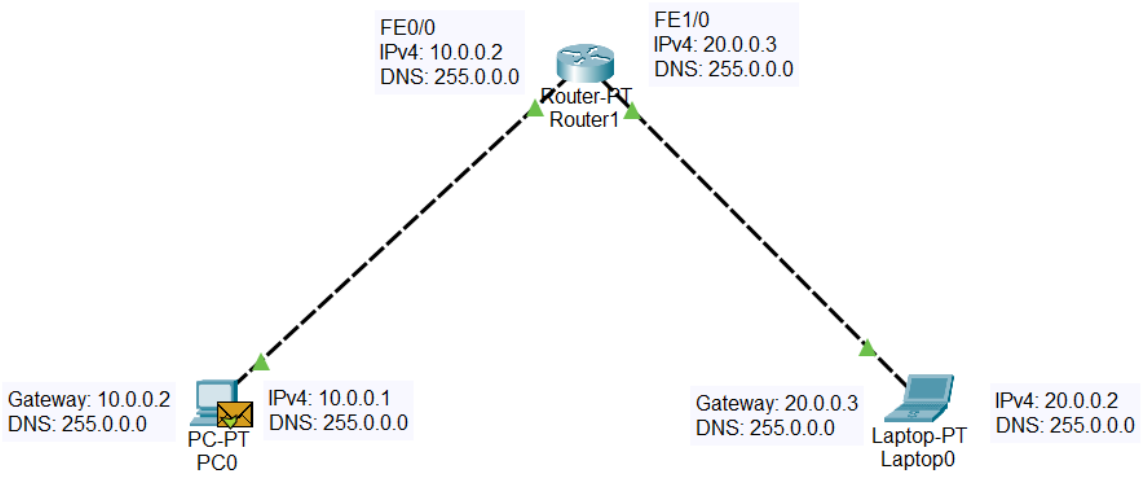
OBSERVATION :

- In simulation mode, on click auto capture/play, the message from source to destination gets traversed through switch0, as this being intellectual, it sends the message only to destination.
- As the destination receives the message, it transmits the signal, indication of transmission successful.
- And switch works on both data link layer and physical layer, due STP-ICMP protocol, it tries to find the shortest path and keep it in memory for further transmission or communication.



LABORATORY PROGRAM – 2

Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	
	In Progress	PC0	Laptop0	ICMP		0.000	N	1	(edit)	
	In Progress	PC0	Laptop0	ICMP		0.000	N	2	(edit)	

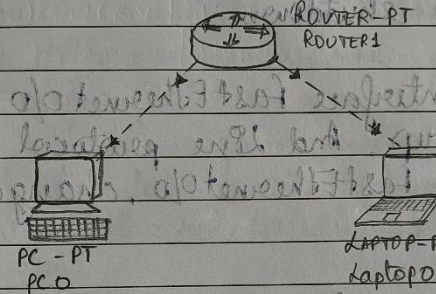
```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 20.0.0.3

Pinging 20.0.0.3 with 32 bytes of data:

Reply from 20.0.0.3: bytes=32 time<1ms TTL=255
Reply from 20.0.0.3: bytes=32 time<1ms TTL=255
Reply from 20.0.0.3: bytes=32 time<1ms TTL=255
Reply from 20.0.0.3: bytes=32 time<1ms TTL=255

Ping statistics for 20.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

8.10.24 DATE: PAGE:

LABORATORY PROGRAM - 2

⇒ CONFIG AIM OF THE EXPERIMENT : Config IP address to routers in packet tracer. Explore the following messages : ping responses, destination unreachable, request timed out, reply.

⇒ TOPOLOGY :



- CONNECTION : Copper Cross-Over

⇒ TOPOLOGY CREATION :

- Using end devices, drag and drop PC-PT and LAPTOP-PT.
- Now, add Router-PT, and connect them using copper cross-over.

DATE: PAGE:

⇒ CONFIGURATION :

⇒ Router : Using CLI, config the left side Interface [PC-PT PC0]
Step 1 : Enable
Step 2 : config t
Step 3 : interface fastethernet 0/0
Step 4 : ip address 10.0.0.2 255.0.0.0
Step 5 : no shutdown

Thus : Interface FastEthernet 0/0, changed state to up. And Line protocol on Interface FastEthernet 0/0, changed state to up.

Similarly config the right side Interface [laptop0]
Step 1 : interface fastethernet 1/0
Step 2 : ip address 20.0.0.3 255.0.0.0
Step 3 : no shutdown :

Thus, Line protocol on Interface FastEthernet 1/0 changed state to up.

⇒ End Devices : Config PC-PT [PC0], IPv4 as 10.0.0.1 and Gateway/DNS IPv4 as 10.0.0.2.

Now, config LAPTOP-PT [laptop0], IPv4 as 20.0.0.2 and Gateway/DNS IPv4 as 20.0.0.3.

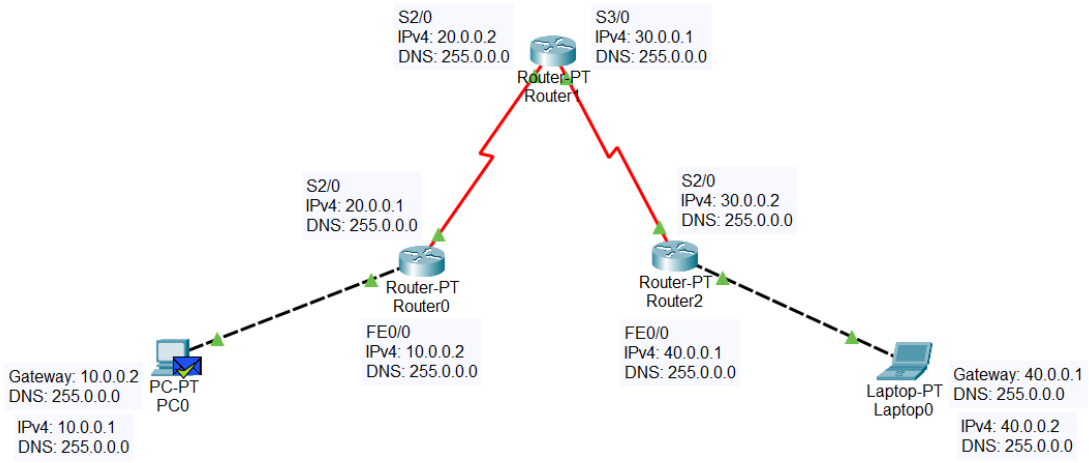
After configuration, lets send a message using ping.
On clicking command prompt" in PC0, type ping 20.0.0.2, at initial, out of 4 packets, 3 were received.

⇒ OBSERVATION :

- When we type ping 20.0.0.2 in PC0 at initial, out of 4 packets, 3 were received at destination and lost (1 packet) And this is due to the initial configuration of 2 different networks, as it identifies the topology.
- Now, if we again prompt" ping 20.0.0.2 in PC0, the loss is 0%. Hence the message transmitted successfully to laptop0.

LABORATORY PROGRAM – 3

Configure static route to the Router.



SHOW IP ROUTE

```
C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    20.0.0.0/8 is directly connected, Serial2/0
S    30.0.0.0/8 [1/0] via 20.0.0.2
S    40.0.0.0/8 [1/0] via 20.0.0.2
```

Figure 3.1: Router0

```
S    10.0.0.0/8 [1/0] via 20.0.0.1
C    20.0.0.0/8 is directly connected, Serial2/0
C    30.0.0.0/8 is directly connected, Serial3/0
S    40.0.0.0/8 [1/0] via 30.0.0.2
```

Figure 3.2: Router1

```
S    10.0.0.0/8 [1/0] via 30.0.0.1
S    20.0.0.0/8 [1/0] via 30.0.0.1
C    30.0.0.0/8 is directly connected, Serial2/0
C    40.0.0.0/8 is directly connected, FastEthernet0/0
```

Figure 3.3: Router3.3

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=36ms TTL=125
Reply from 40.0.0.2: bytes=32 time=34ms TTL=125
Reply from 40.0.0.2: bytes=32 time=30ms TTL=125
Reply from 40.0.0.2: bytes=32 time=26ms TTL=125

Ping statistics for 40.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 26ms, Maximum = 36ms, Average = 31ms
```

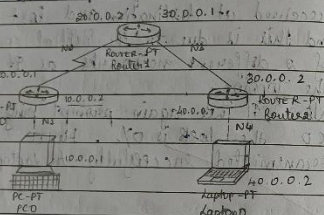
8.10.24

DATE: PAGE:

LABORATORY PROGRAM - 3

⇒ AIM OF THE EXPERIMENT : To demonstrate the configuration of default routes to the router.

⇒ TOPOLOGY :



⇒ TOPOLOGY CREATION :

- Using end devices, drag and drop PC-PT and Laptop-PT.
- Also, add 3 routers connected as shown above.
- Connection : Copper cross-over (From End device to router)
  (Connection : Serial DTE (From Router to router)

ip route destination network ip destination mask

DATE: PAGE:

⇒ Configuration :

⇒ Router : ip address 20.0.0.1 255.0.0.0 (Serial0)
- Config Router0 : ip address 10.0.0.2 255.0.0.0 (FE0/0)
  Routing : ip route 30.0.0.0 255.0.0.0 20.0.0.2
  ip route 40.0.0.0 255.0.0.0 20.0.0.2
- Config Router1 : ip address 20.0.0.2 255.0.0.0 (Serial2/0)
  ip address 30.0.0.1 255.0.0.0 (Serial3/0)
  Routing : ip route 10.0.0.0 255.0.0.0 20.0.0.1
  ip route 40.0.0.0 255.0.0.0 30.0.0.2
- Config Router2 : ip address 30.0.0.2 255.0.0.0 (Serial2/0)
  ip address 40.0.0.1 255.0.0.0 (FE0/0)
  Routing : ip route 20.0.0.0 255.0.0.0 30.0.0.1
  ip route 10.0.0.0 255.0.0.0 30.0.0.1

Thus, configuired.

⇒ End Devices :
- Config PC-PT [PC0] : ip address 10.0.0.1 255.0.0.0 and Gateway : 10.0.0.2 , DNS : 255.0.0.0
- Config Laptop-PT [Laptop0] : ip address 40.0.0.2 255.0.0.0 and Gateway : 40.0.0.1 , DNS : 255.0.0.0

Thus, configuration is successful. Now lets send message using ping-
In PC-0 , ping 40.0.0.2 .
Pinging 40.0.0.2 with 32 bytes of data :
Reply from 40.0.0.2 : bytes=32 time=36ms TTL=125
Reply from 40.0.0.2 : bytes=32 time=34ms TTL=125
Reply from 40.0.0.2 : bytes=32 time=30ms TTL=125
Reply from 40.0.0.2 : bytes=32 time=26ms TTL=125

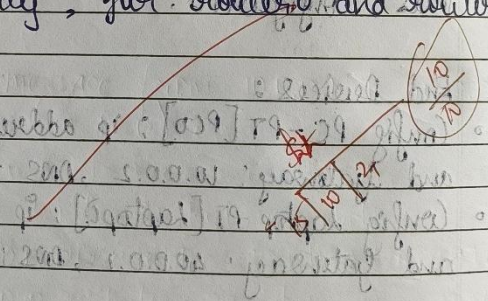
DATE: PAGE:

OBSERVATIONS :

Ping statistics for 40.0.0.2 :
Packets : Sent = 4, received = 4,
lost = 0 (0% loss)
And approx round trip times in ms:
Min = 28ms, Max = 36 ms, Avg = 31 ms.

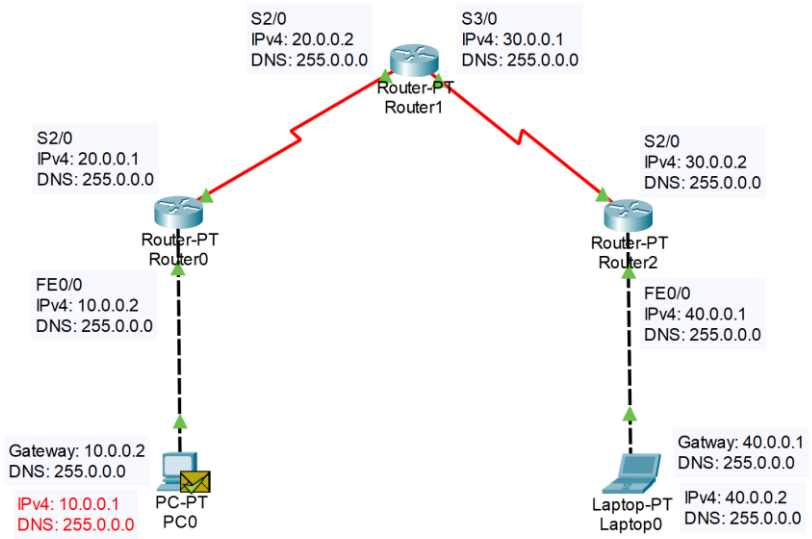
And after configuration in routers :
> show ip route
S 10.0.0.0/8 [1/0] via 20.0.0.1
C 20.0.0.0/8 is directly connected, serial 2/0
C 30.0.0.0/8 is directly connected, serial 3/0
S 40.0.0.0/8 [1/0] via 30.0.0.2

Also, similarly, for router0 and router2, are displayed.



LABORATORY PROGRAM – 4(A)

Configure default route, static route to the Router.



SHOW IP ROUTE

```
Gateway of last resort is 20.0.0.2 to network 0.0.0.0

C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    20.0.0.0/8 is directly connected, Serial2/0
S*   0.0.0.0/0 [1/0] via 20.0.0.2
```

Figure 4.1: Router0

```
S    10.0.0.0/8 [1/0] via 20.0.0.1
C    20.0.0.0/8 is directly connected, Serial2/0
C    30.0.0.0/8 is directly connected, Serial3/0
S    40.0.0.0/8 [1/0] via 30.0.0.2
```

Figure4.2: Router1

```
Gateway of last resort is 30.0.0.1 to network 0.0.0.0

C    30.0.0.0/8 is directly connected, Serial2/0
C    40.0.0.0/8 is directly connected, FastEthernet0/0
S*   0.0.0.0/0 [1/0] via 30.0.0.1
```

Figure 4.3: Router2

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	

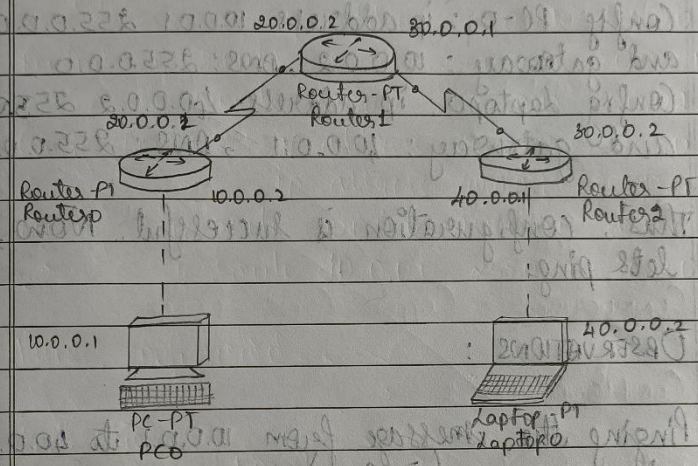
```
C:\>ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=34ms TTL=125
Reply from 40.0.0.2: bytes=32 time=33ms TTL=125
Reply from 40.0.0.2: bytes=32 time=30ms TTL=125
Reply from 40.0.0.2: bytes=32 time=33ms TTL=125

Ping statistics for 40.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 30ms, Maximum = 34ms, Average = 32ms
```

22.10.24 DATE: PAGE:

LABORATORY PROGRAM - 4

⇒ AIM OF THE EXPERIMENT : TO CONFIGURE default and static routes to a connection of routers.

⇒ TOPOLOGY :



CONNECTION : Copper Cross-Over (from end devices to routers)
Serial DTE (from Router to router)

⇒ CONFIGURATION :

→ Router :

- Config Router0 : ip address 20.0.0.1 255.0.0.0 (S2/0)
  ip address 10.0.0.2 255.0.0.0 (FE0/0)
  Default Route : ip route 0.0.0.0 0.0.0.0 20.0.0.2
- Config Router1 : ip address 20.0.0.2 255.0.0.0 (S2/0)
  ip address 30.0.0.1 255.0.0.0 (S3/0)

DATE: PAGE:

Static Route : ip route 10.0.0.0 255.0.0.0 20.0.0.1
ip route 40.0.0.0 255.0.0.0 30.0.0.2

- Config Router2: ip address 30.0.0.2 255.0.0.0 (S2/0)
  ip address 40.0.0.1 255.0.0.0 (FE0/0)

Default Route: ip route 0.0.0.0 0.0.0.0 30.0.0.1

→ End Devices:

- Config PC-0 : ip address 10.0.0.1 255.0.0.0, and gateway : 10.0.0.2 , DNS: 255.0.0.0
- Config Laptop0: ip address 40.0.0.2 255.0.0.0, and gateway: 40.0.0.1 , DNS : 255.0.0.0

Thus, configuration is successful. Now, lets ping.

↝ OBSERVATIONS :

- Pinging the message from 10.0.0.1 to 40.0.0.2
  > ping 40.0.0.2 (in PC0)

Pinging 40.0.0.2 with 32 bytes of data:
Reply from 40.0.0.2 : bytes=32 time=9ms TTL=125
Reply from 40.0.0.2 : bytes=32 time=8ms TTL=125
Reply from 40.0.0.2 : bytes=32 time=5ms TTL=125
Reply from 40.0.0.2 : bytes=32 time=10ms TTL=125

Ping statistics for 40.0.0.2 :
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 5ms, Maximum = 10ms, Average = 8ms

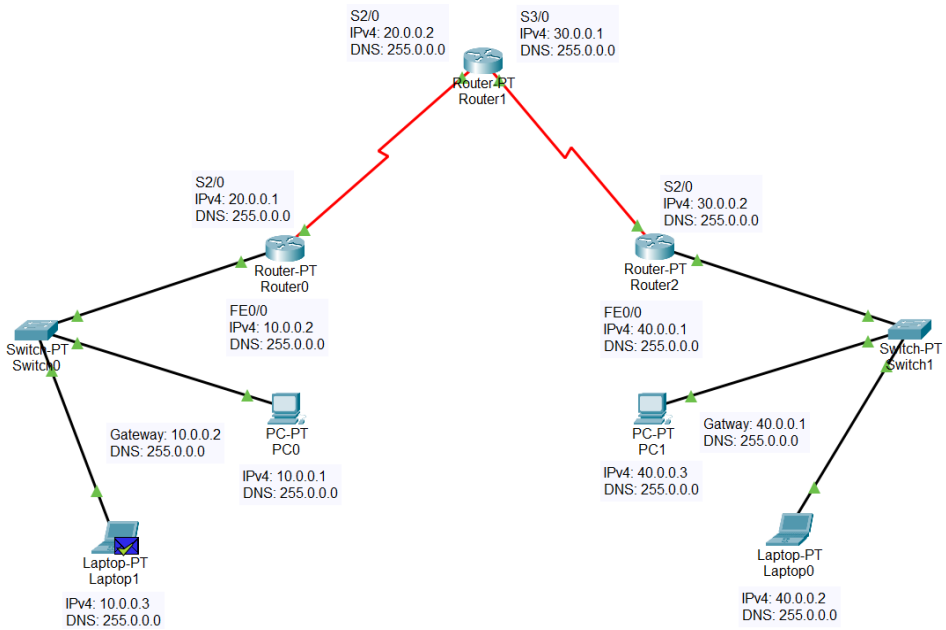
DATE: PAGE:

And after configuration, in router0 :
> show ip route
C 10.0.0.0/8 is directly connected, FastEthernet0/0
C 30.0.0.0/8 is directly connected, Serial2/0
S* 0.0.0.0/0 [1/0] via 30.0.0.2

And, similarly, for router1 and router2 are displayed.

LABORATORY PROGRAM – 4(B)

Configure default route, static route to the Router, inclusive switches.



SHOW IP ROUTE

```
Gateway of last resort is 20.0.0.2 to network 0.0.0.0

C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    20.0.0.0/8 is directly connected, Serial2/0
S*   0.0.0.0/0 [1/0] via 20.0.0.2
```

Figure 4.1: Router0

```
S    10.0.0.0/8 [1/0] via 20.0.0.1
C    20.0.0.0/8 is directly connected, Serial2/0
C    30.0.0.0/8 is directly connected, Serial3/0
S    40.0.0.0/8 [1/0] via 30.0.0.2
```

Figure4.2: Router1

```
Gateway of last resort is 30.0.0.1 to network 0.0.0.0

C    30.0.0.0/8 is directly connected, Serial2/0
C    40.0.0.0/8 is directly connected, FastEthernet0/0
S*   0.0.0.0/0 [1/0] via 30.0.0.1
```

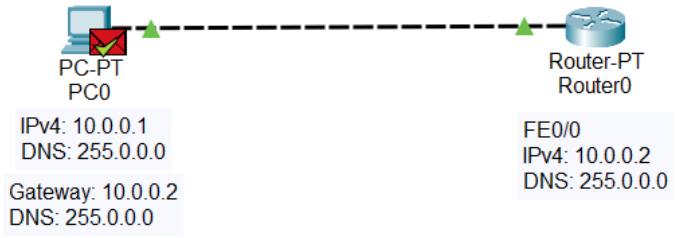
Figure 4.3: Router2

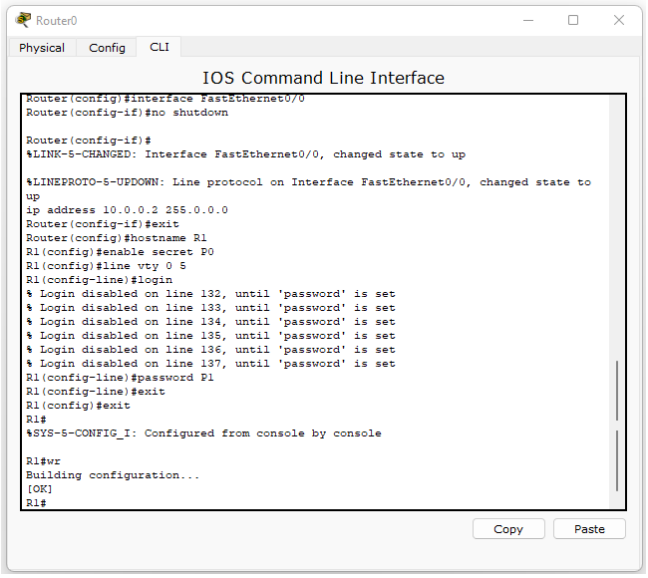
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	

```
C:\>ping 40.0.0.3

Pinging 40.0.0.3 with 32 bytes of data:

Reply from 40.0.0.3: bytes=32 time=35ms TTL=125
Reply from 40.0.0.3: bytes=32 time=37ms TTL=125
Reply from 40.0.0.3: bytes=32 time=24ms TTL=125
Reply from 40.0.0.3: bytes=32 time=38ms TTL=125

Ping statistics for 40.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 24ms, Maximum = 38ms, Average = 33ms
```

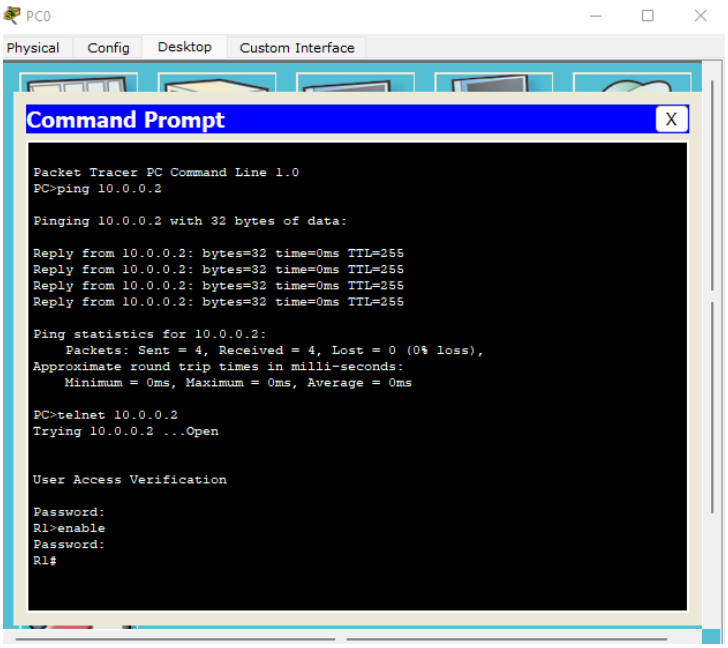
LABORATORY PROGRAM – 5

To understand the operation of TELNET by accessing the router in server room from a PC in IT office.





Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Router0	ICMP		0.000	N	0	(edit)	



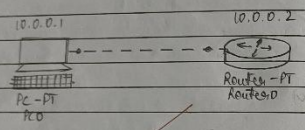
DATE: PAGE:

29.10.24

LABORATORY EXPERIMENT - 5

AIM OF THE EXPERIMENT : To understand the operation of TELNET by accessing the router in the server room from a PC in IT office.

TOPOLOGY :



CONNECTION : Copper Cross - Over.

CONFIGURATION :

- After topology creation
- Config PC0 and Router0 with corresponding address
- Router0 Config : Ip address 10.0.0.2 255.0.0.0
- PC0 config : Ip address 10.0.0.1 255.0.0.0
  And gateway as : 10.0.0.2 255.0.0.0.
- And Configuring the config of Router0.

```
# hostname R1
# enable secret P0
# line vty 0 5
# login
% Login disabled on line 132, until 'password' is set
% Login disabled on line 137, until 'password' is set
# line vty 0 5 [Virtual Terminal is allowed for line 0 1 2 3 4 5]
# login
```

DATE: PAGE:

```
# password P1
R1# wr [Building configuration]
```

TOPOLOGY CREATION :

- Make the connection with PC and router and copper - cross over.

OBSERVATION :

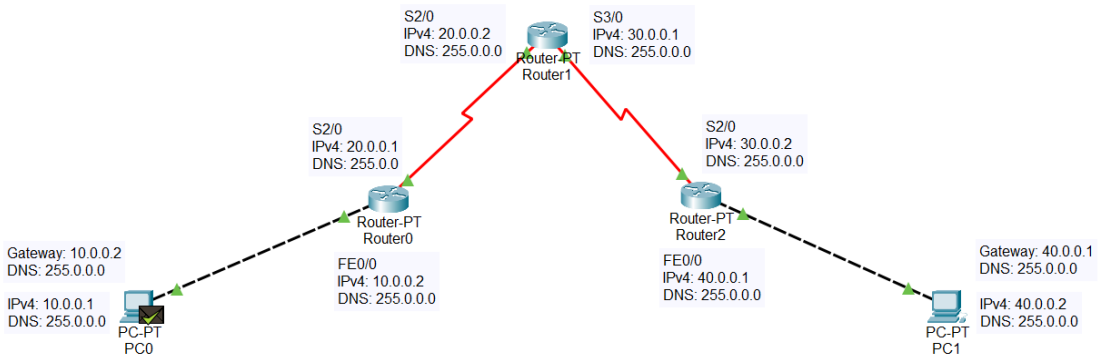
- After configuration, pinging message from PC0 to R1.

```
> ping 10.0.0.2.
Pinging 10.0.0.2 with 32 bytes of data :
Reply from 10.0.0.2 : bytes = 32 time = 0ms TTL = 255
:
Reply from 10.0.0.2 : bytes = 32 time = 0ms TTL = 255
Ping statistics for 10.0.0.2 :
    Packets : Sent = 4, Recieved = 4, Lost = 0 (0% Loss),
Approximate round trip times in milli-seconds :
    Min = 0ms, Max = 0ms, Avg = 0ms
PC> telnet 10.0.0.2
Trying 10.0.0.2 ... Open
User Access Verification
Password : P1
R1> enable
Password : P0
R1#
```

Now, we care able to access R1 and make configuration or change

LABORATORY PROGRAM – 6

Demonstrate the TTL/ Life of a Packet.



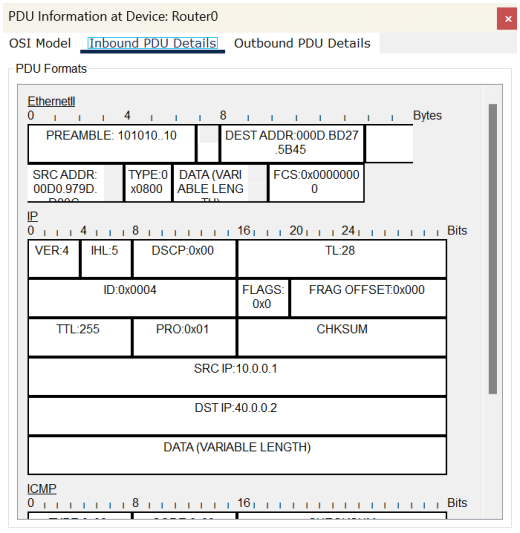


Figure 6.1: Inbound PDU, Router0

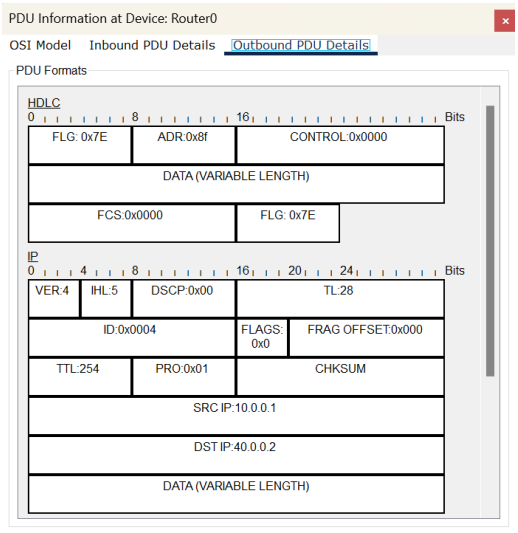


Figure 6.2: Outbound PDU, Router0

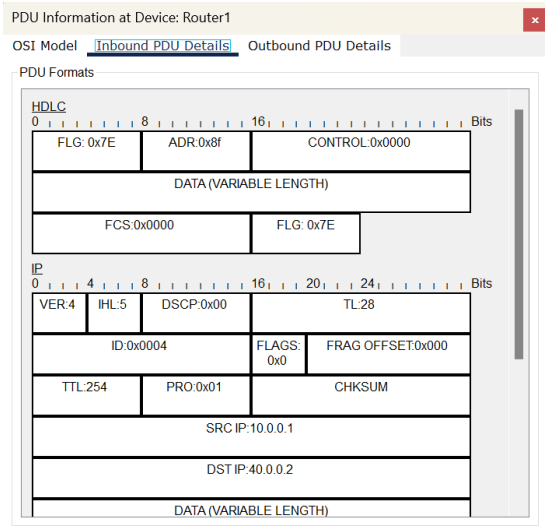


Figure 6.3: Inbound PDU, Router1

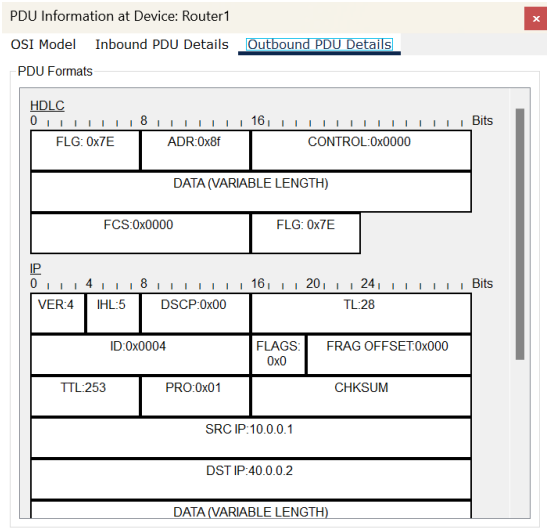


Figure 6.4: Outbound PDU, Router1

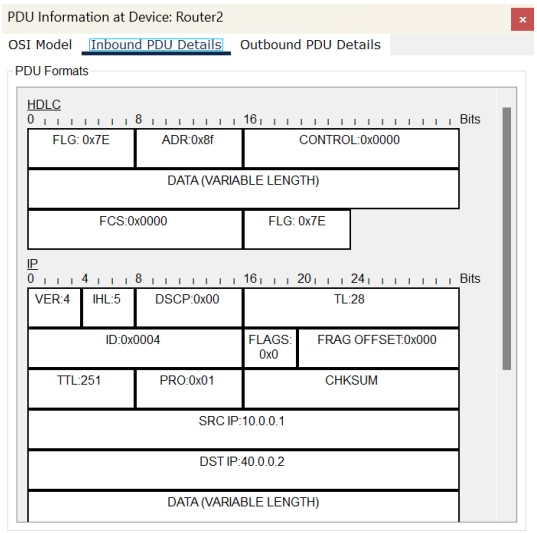


Figure 6.5: Inbound PDU, Router2

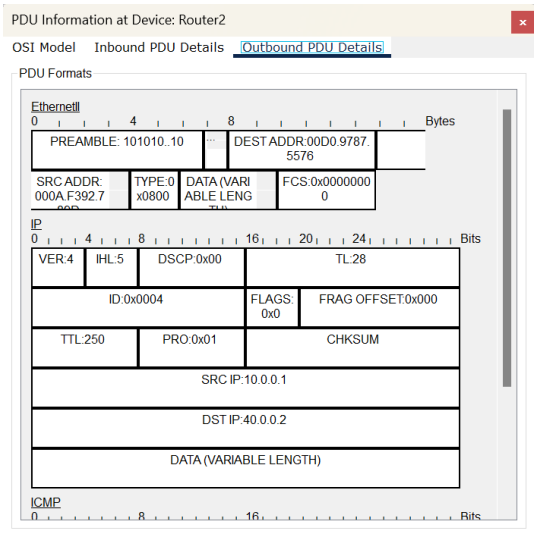


Figure 6.6: Outbound PDU, Router2

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)	

```
C:\>ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=72ms TTL=123
Reply from 40.0.0.2: bytes=32 time=53ms TTL=123
Reply from 40.0.0.2: bytes=32 time=55ms TTL=123
Reply from 40.0.0.2: bytes=32 time=69ms TTL=123

Ping statistics for 40.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 53ms, Maximum = 72ms, Average = 62ms
```

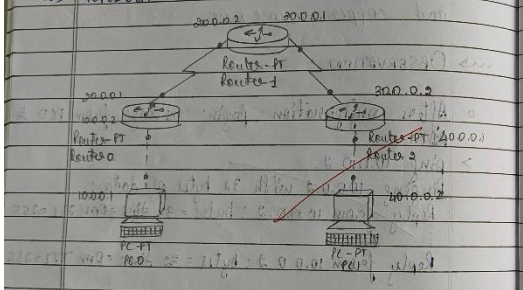
DATE: PAGE:

29-10-24

Laboratory Program - 6

Aim Of The Experiment : Demonstrate the TTL / life of a packet.

Topology :



Connections : Copper Cross - Over (from end devices to routers)
Serial DTE (from Router to Router)

Topology Creation :

Construct a network as mentioned in diagram.

Configuration :

- Config all routers and end devices accordingly as done in previous experiment with default and static routing.

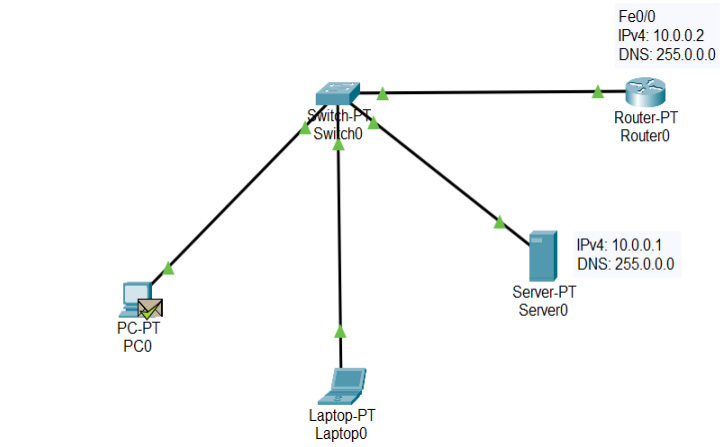
DATE: PAGE:

Observation :

- After configuration and pinging message from 10.0.0.1 to 10.0.0.2 using Simple PDU.
- In simulation mode, on clicking PDU at Initial stage, at Inbound PDU details the TTL : 255 and after 1 hop, the Outbound PDU details on TTL : 254. Thus, at each hop, the TTL is decreased by one.
- At destination, in PC1, the TTL is 250, means, there is a decreament of TTL at each hop.
- The message is transmitted successfully
- And if the TTL is less than requiured hop, the packet is discharded, and again a new message has to be resent.

LABORATORY PROGRAM – 7(A)

To Configure IP addresses of the host using DHCP server within a LAN.



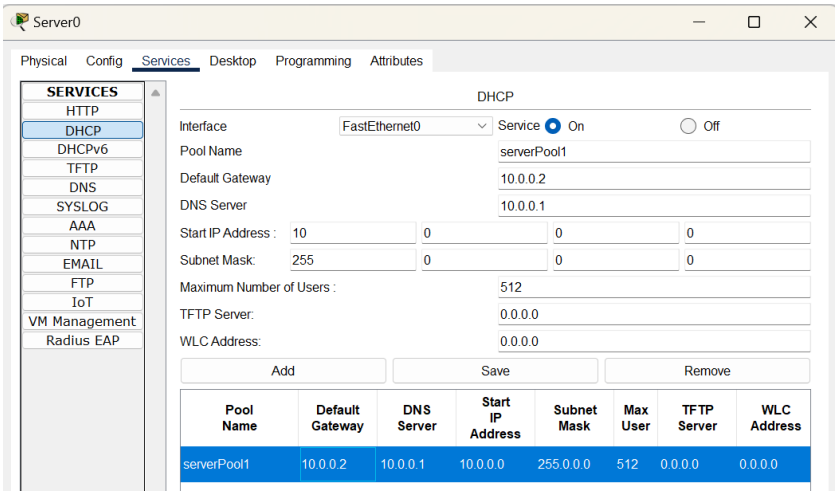


Figure 7.1: DHCP Service, Server0

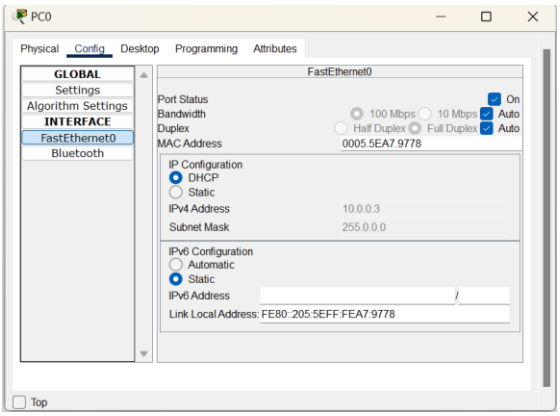


Figure 7.2: DHCP Service, PC0

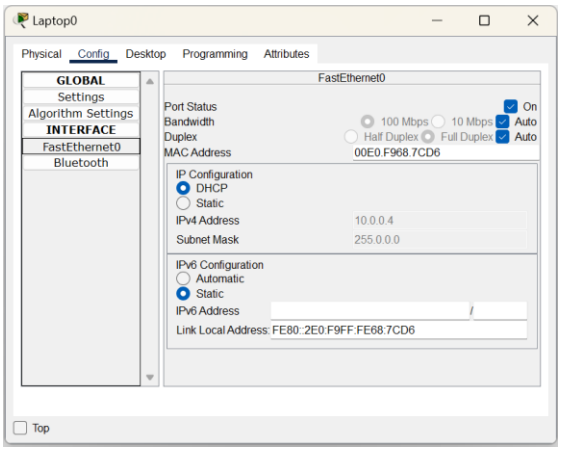
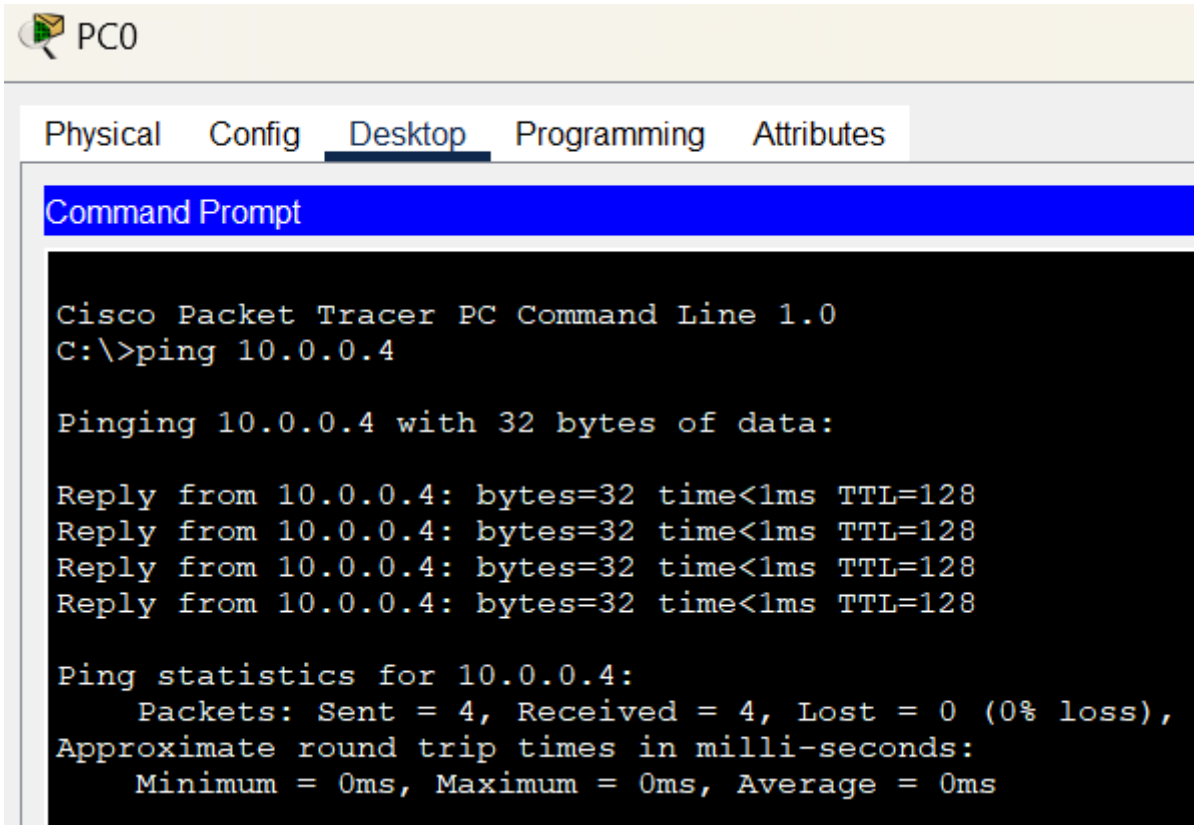


Figure 7.3: DHCP Service, Laptop0

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	



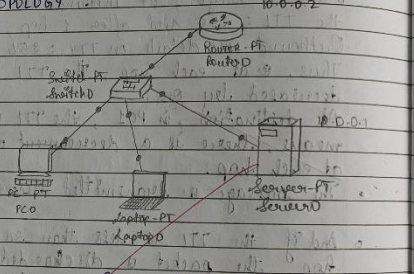
DATE: PAGE:

12-11-24

LABORATORY PROGRAM - 7(*)

⇒ AIM OF THE EXPERIMENT: To configure IP addresses of the host using DHCP server present within the same LAN.

⇒ TOPOLOGY :



CONNECTIONS: Copper Straight-Through.

⇒ CONFIGIRATION :

- Connect the topology using copper straight through
- Server-PT Config:
  At FastEthernet0 : IP address : 10.0.0.1
  Subnet Mask : 255.0.0.0
  Gateway/DNS : Gateway : 10.0.0.2
  DNS Server : 10.0.0.1
  At Services : Click DHCP protocol :
  On the service
  PoolName : serverpool1

DATE: PAGE:

Default Gateway : 10.0.0.2
DNS Server : 10.0.0.1
And save the DHCP.

At Router-PT : In FastEthernet0/0.
IP address : 10.0.0.2
Port status : On.
Thus all the configuration are done.

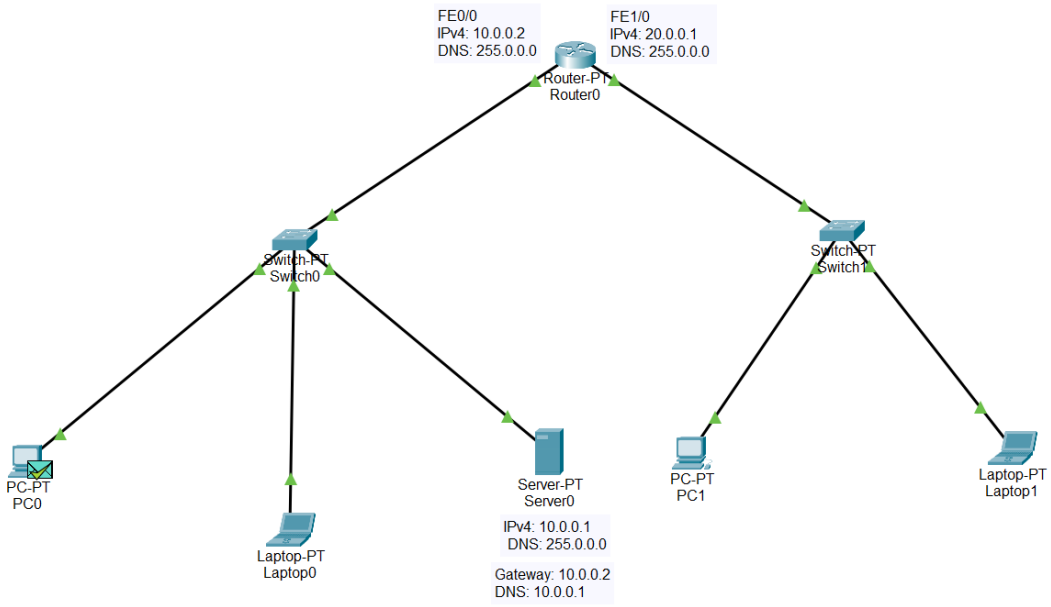
⇒ OBSERVATION :

Now, for the PC0 and LAPTOP0, the IP addresses are assigned automatically through DHCP protocol.
On clicking Gateway/DNS, the DHCP,
→ the PC0 : FastEthernet0: IP Address : 10.0.0.4
→ Laptop0 : FastEthernet0 : IP Address : 10.0.0.3.

- In PC0 > ping 10.0.0.4
  Pinging 10.0.0.4 with 32 bytes of data:
  Reply from 10.0.0.4 : bytes = 32 time = 1ms TTL = 128
  Reply from 10.0.0.4 : bytes = 32 time = 10ms TTL = 128
  Reply from 10.0.0.4 : bytes = 32 time = 1ms TTL = 128
  Reply from 10.0.0.4 : bytes = 32 time = 7ms TTL = 128
  Ping Statistics for 10.0.0.4:
  Packets : Sent = 4, Recieved = 4, Lost = 0 (0% loss)
  Approx round trip times in milli-seconds:
  Min = 1ms, Max = 10ms, Average = 4ms

LABORATORY PROGRAM – 7(B)

To Configure IP addresses of the host using DHCP server outside a LAN.



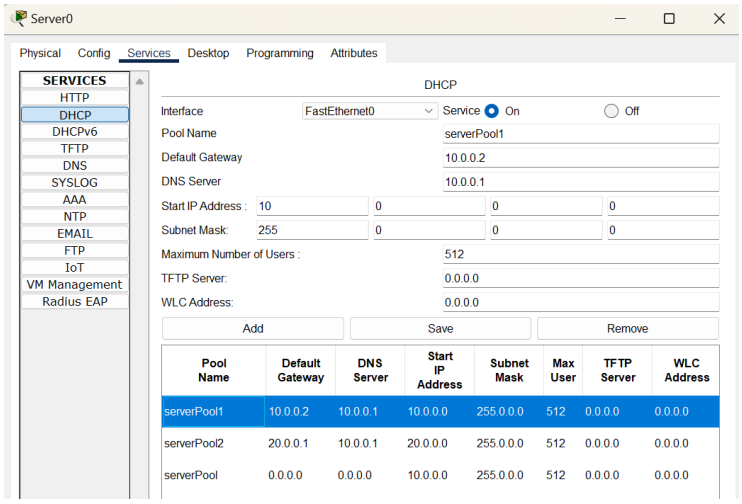


Figure 7.2.1: DHCP Service, Server0

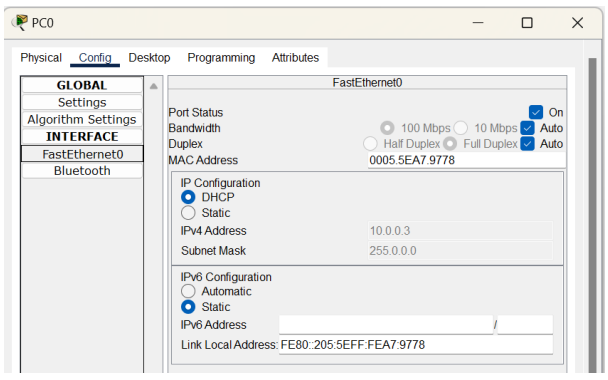


Figure7.2.2: DHCP Service, PC0

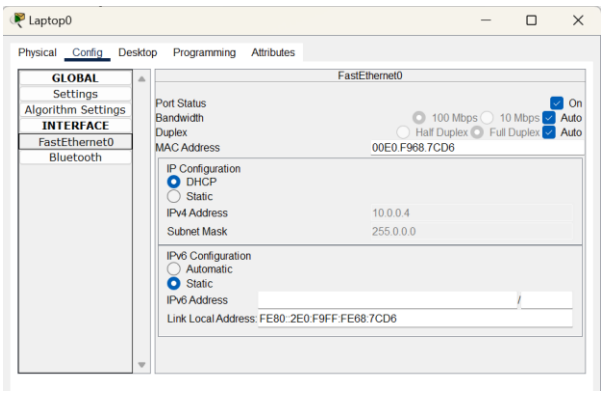


Figure 7.2.3: DHCP Service, Laptop0

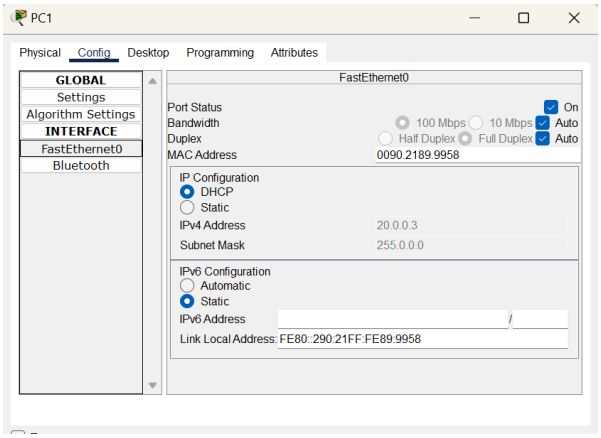


Figure 7.2.4: DHCP Service, PC1

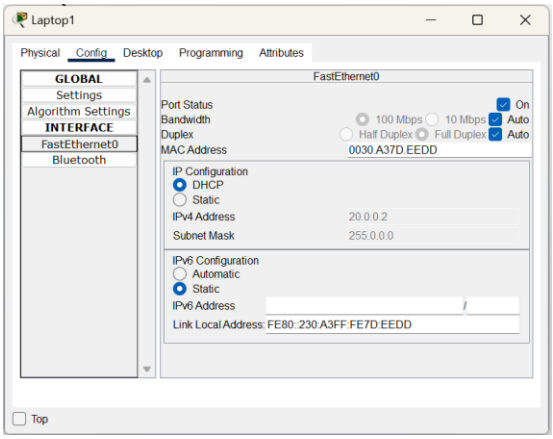
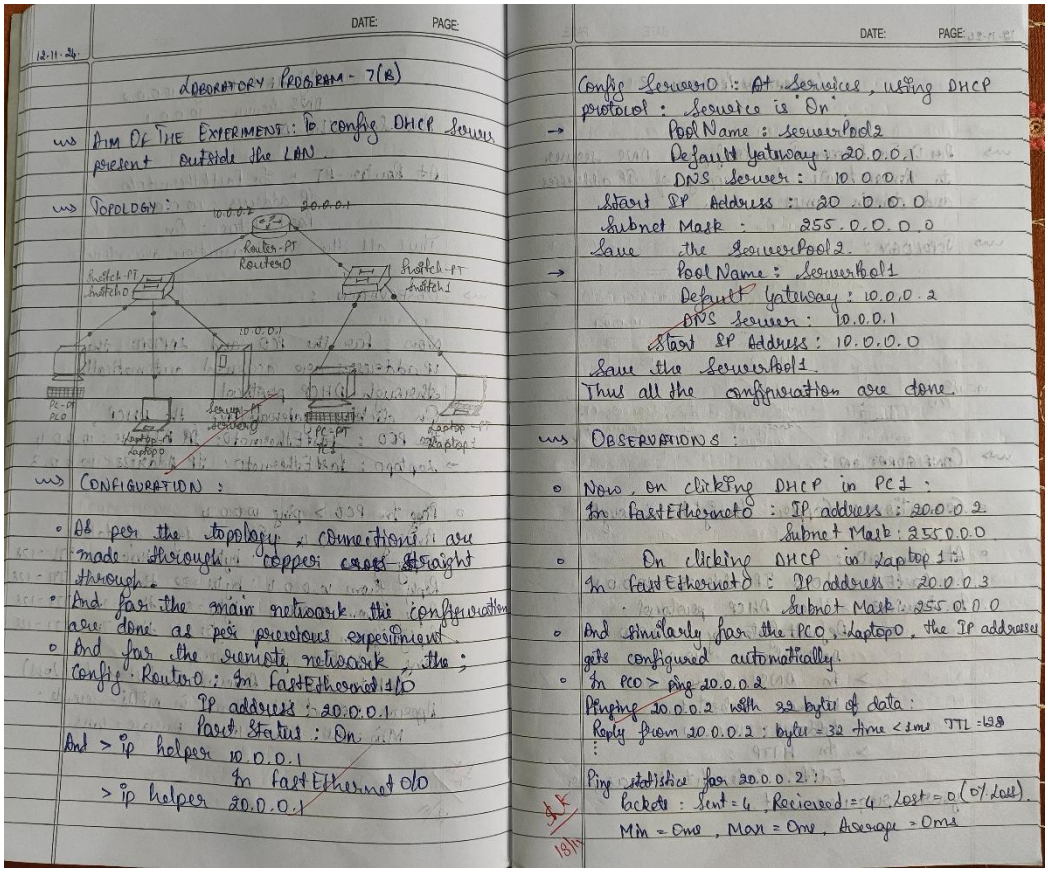


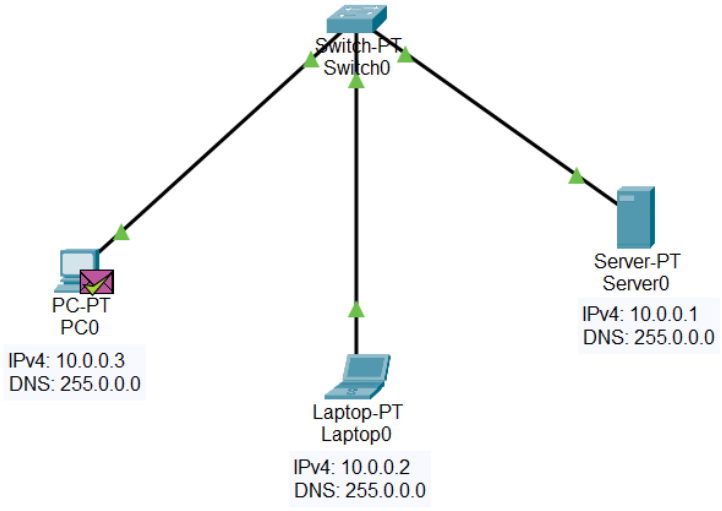
Figure 7.2.5: DHCP Service, Laptop1

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	
	Successful	PC1	Laptop1	ICMP		0.004	N	1	(edit)	



LABORATORY PROGRAM – 8

To Configure DNS server to demonstrate the mapping of IP addresses and Domain names.



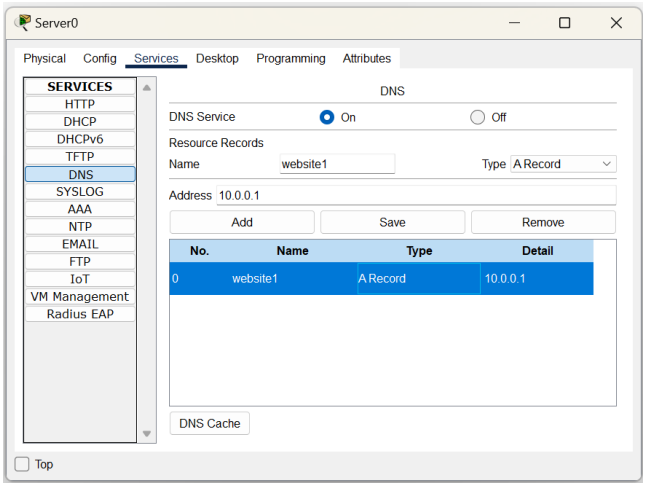


Figure 8.1: DNS Service, Server0

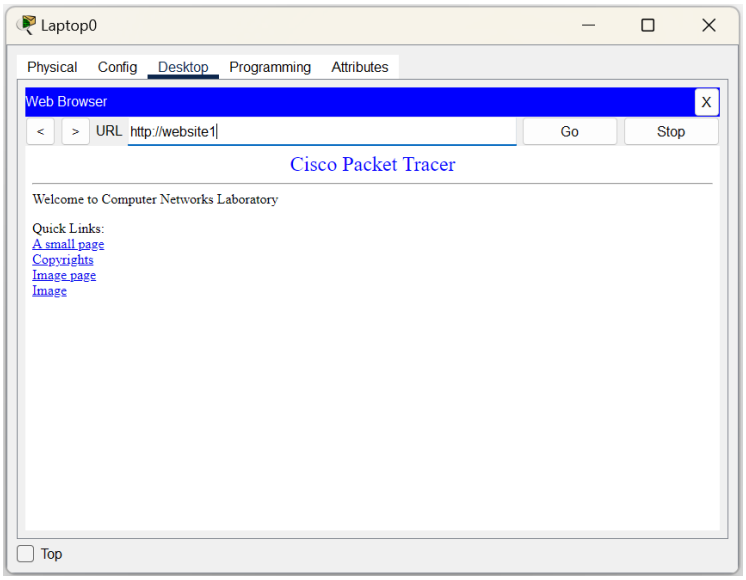


Figure 8.2: DNS Service, Laptop0

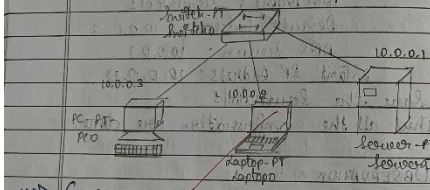
DATE: PAGE:

18.11.24

LABORATORY PROGRAM - 8

AIM OF THE EXPERIMENT: To configure DNS server to demonstrate the mapping of IP addresses and domain names.

TOPOLOGY:



CONFIGURATION:

- Connection: Copper Straight through
- At Server0:
  - In FastEthernet0: IP address: 10.0.0.1
  - In Services > Use DHCP protocol:
    - Default Gateway: 0.0.0.0
    - DNS Service: 10.0.0.1
  - > In DNS Services:
    - Name: webs8to1
    - Address: 10.0.0.1
  - > In HTTP:
    - Edit index.html, as per your requirement.
- In PC0: Click on DHCP, the IP address get configured automatically.
- In Laptop0: On clicking once, the IP address get configured automatically.

Thus all the configuration are done.

OBSERVATION:

- Using any end device, click on "web browser" in "Desktop" menu.
- There, on typing webs8to1, the domain name, index.html gets displayed.

URL: http://webs8to1

Cisco Packet Tracer

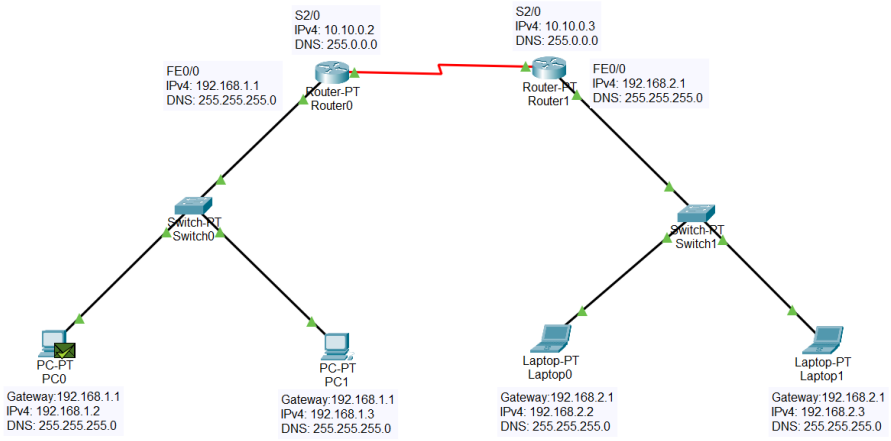
Welcome to Computer Networks Laboratory

Quick Links:
A small page
Copyrights
Image page
Image

25
18/11

LABORATORY PROGRAM – 9

To Configure RIP routing protocol in Routers.



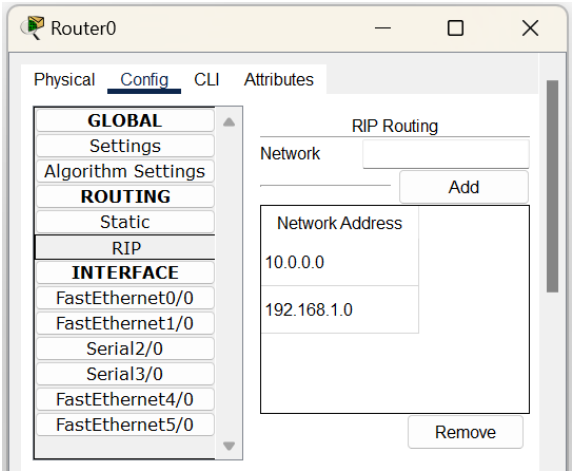


Figure 9.1: RIP, Router0

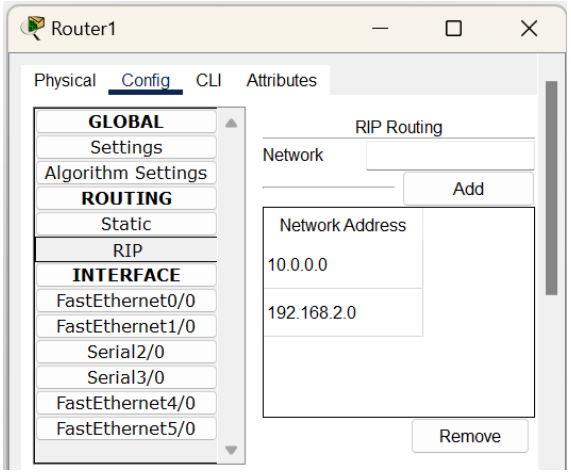


Figure 9.2: RIP, Router

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop1	ICMP		0.000	N	0	(edit)	

```
C:\>ping 192.168.2.3

Pinging 192.168.2.3 with 32 bytes of data:

Reply from 192.168.2.3: bytes=32 time=18ms TTL=126
Reply from 192.168.2.3: bytes=32 time=14ms TTL=126
Reply from 192.168.2.3: bytes=32 time=1ms TTL=126
Reply from 192.168.2.3: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.2.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 18ms, Average = 8ms
```

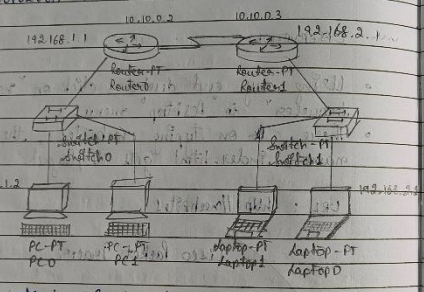
DATE: PAGE:

19.11.24

LABORATORY PROGRAM - 9

⇒ AIM OF THE EXPERIMENT: To configure RIP routing protocol in Routers.

⇒ TOPOLOGY :



- Connection : Copper straight-through and Serial DTE.

⇒ CONFIGURATION :

- Construct the topology as given above and use corresponding connections.
- Now, lets config all end devices.
- PC0 :
  > IP address : 192.168.1.2
  Subnet Mask : 255.255.255.0
  Gateway : 192.168.1.1

DATE: PAGE:

- Laptop0 :
  > IP address : 192.168.2.2
  > Subnet Mask : 255.255.255.0
  > Gateway : 192.168.2.1

- Router0 :
  In FE0/0 : IP address : 192.168.1.1
  Subnet Mask : 255.255.255.0
  In Serial2/0 : IP address : 10.10.0.2
  Subnet Mask : 255.0.0.0
  Clock Rate : 64000

RIP Routing :
Router(config) # router rip
Router(config-router) # network 192.168.1.0
Router(config-router) # network 10.0.0.0
Router(config-router) # end

- Router1 :
  In FE0/0 : IP address : 192.168.2.1
  Subnet Mask : 255.255.255.0
  In Serial2/0 : IP address : 10.10.0.3
  Subnet Mask : 255.0.0.0
  Clock Rate : 2000000

RIP Routing :
router rip
network 192.168.2.0
network 10.0.0.0
end.

Thus all the configuration is done.

⇒ OBSERVATION :

- RIP, used to help routers exchange info about the best paths to route network traffic.
- Thus, after using RIP routing, the transmission of messages are possible.
- On pinging PDU from PC0 to Laptop0, the transmission is successful.

- PC0 > ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

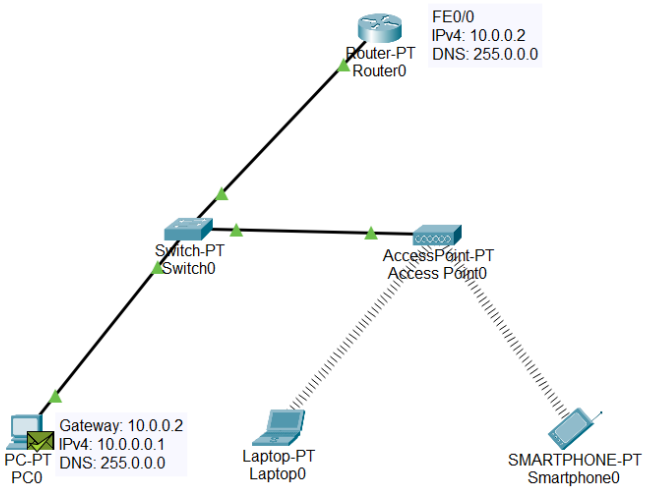
Reply from 192.168.2.2 : bytes = 32 time = 7ms TTL = 126
Reply from 192.168.2.2 : bytes = 32 time = 7ms TTL = 126
Reply from 192.168.2.2 : bytes = 32 time = 6ms TTL = 126
Reply from 192.168.2.2 : bytes = 32 time = 6ms TTL = 126

Ping statistics for 192.168.2.2 :
Packets : Sent = 4, Received = 4, Lost = 0 (0% loss),
Approx round trip times in milli-seconds :
Min = 6ms, Max = 7ms, Avg = 6ms

18/11/24

LABORATORY PROGRAM – 10

To demonstrate communication between two devices using a wireless LAN.



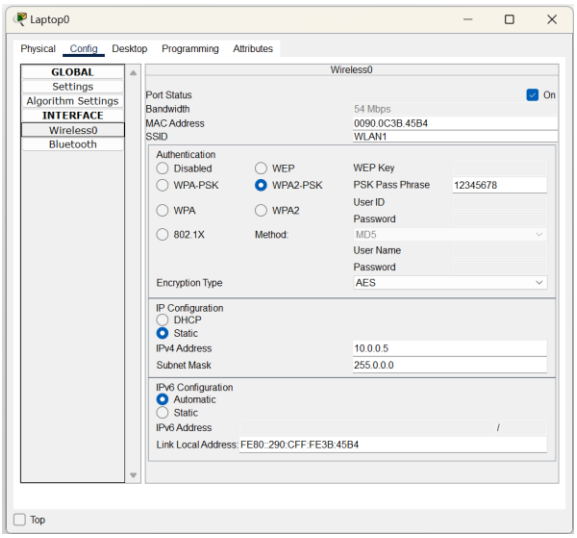


Figure 10.1: Laptop0, Wireless0

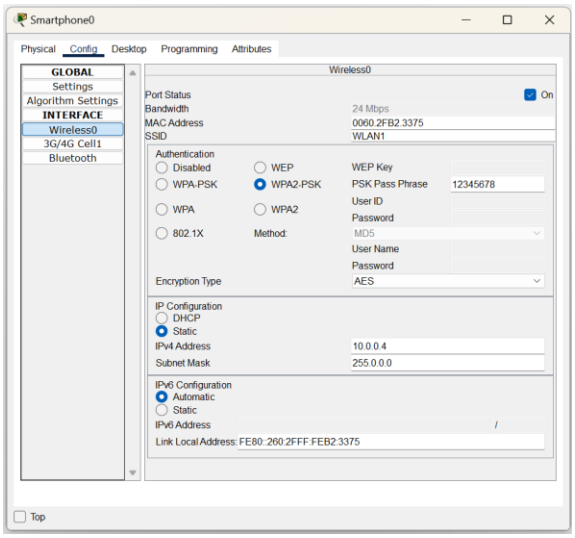
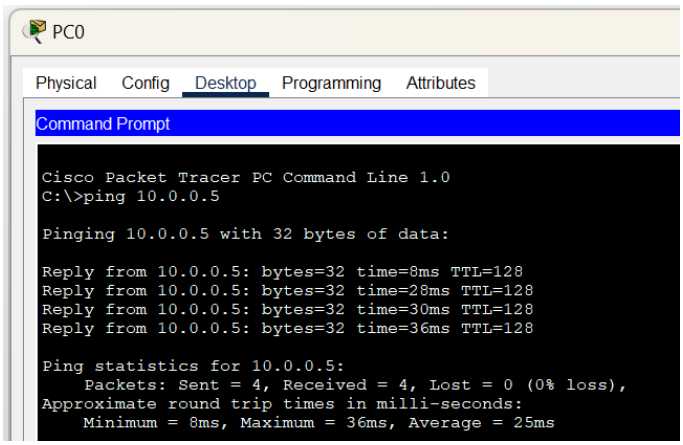


Figure 10.2: Smartphone0, Wireless0

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	

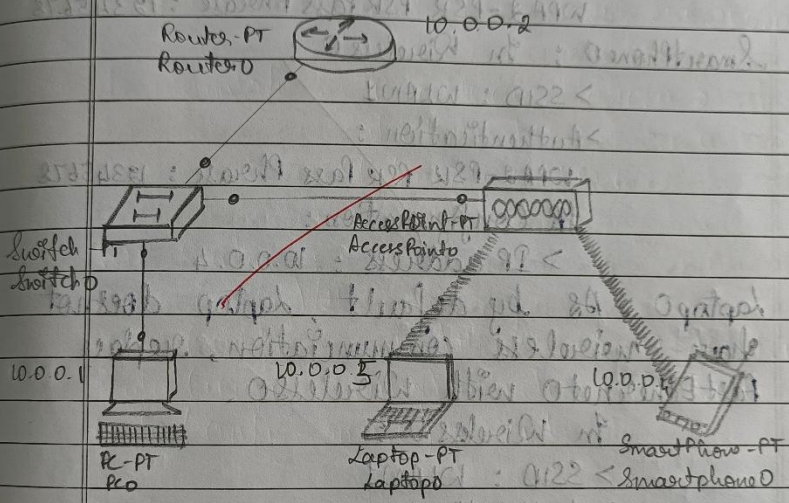


26.11.24 DATE: PAGE:

LABORATORY PROGRAM - 10

⇒ AIM OF THE EXPERIMENT : To demonstrate communication between two devices using a wireless LAN

⇒ TOPOLOGY :



⇒ CONFIGURATION :

- Construct the topology as given above and use connection : Copper Straight-through and wireless connection from accesspoint0 to laptop0 and smartphone0.
- Now, configuration of devices :

1. Router0 : In FastEthernet0/0
   > ip address : 10.0.0.2
2. PC0 : In FE0/0
   > IP address : 10.0.0.1
   > Gateway : 10.0.0.2

DATE: PAGE:

3. AccessPoint0 : In PORT0 :
   > Bandwidth : Auto
   > Duplex : Auto
   In PORT1 :
   In PORT1 :
   > SSID : WLAN1
   > Authentication :
   WPA2-PSK PSK Pass Phrase : 12345678
4. Smartphone0 : In Wireless0 :
   > SSID : WLAN1
   > Authentication :
   WPA2-PSK PSK Pass Phrase : 12345678
   IP Configuration :
   > IP address : 10.0.0.4
5. Laptop0 : As by default, Laptop doesnot have wireless communication, replace FastEthernet0 with Wireless0.
   In Wireless0 :
   > SSID : WLAN1
   > Authentication :
   WPA2-PSK PSK Pass Phrase : 12345678
   > IP address : 10.0.0.5

Thus all the configuration are done.

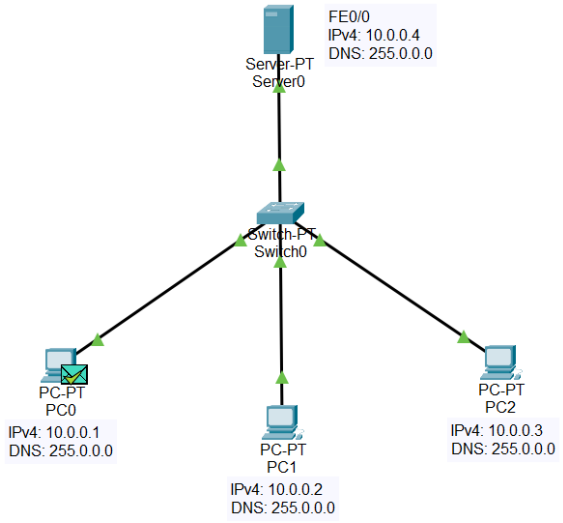
DATE: PAGE:

OBSERVATION :

- As we successful established wireless LAN (communication), now it is possible to ping messages from any devices.
- Pinging message from Laptop0 to Smartphone0
  > ping 10.0.0.4
  Pinging 10.0.0.4 with 32 bytes of data :
  Reply from 10.0.0.4 : bytes = 32 time = 31ms TTL = 128
  Reply from 10.0.0.4 : bytes = 32 time = 17ms TTL = 128
  Reply from 10.0.0.4 : bytes = 32 time = 23ms TTL = 128
  Reply from 10.0.0.4 : bytes = 32 time = 18ms TTL = 128
  Ping Statistics for 10.0.0.4 :
  Packets : Sent = 4, Received = 4, Lost = 0 (0% loss)
  Approx round trip times in ms :
  Min = 17ms, Max = 31ms, Avg = 22ms

LABORATORY PROGRAM – 11

To demonstrate the working of Address Resolution Protocol (ARP) within a LAN for communication.



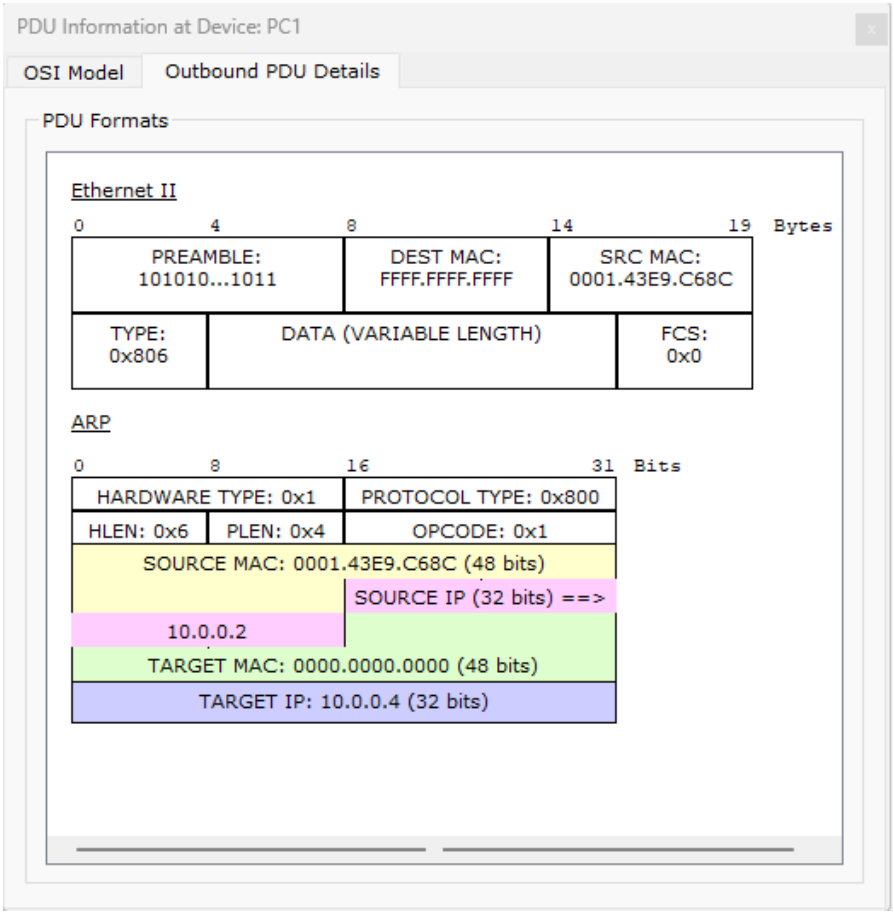


Figure 11.1: Inbound ARP, PC1

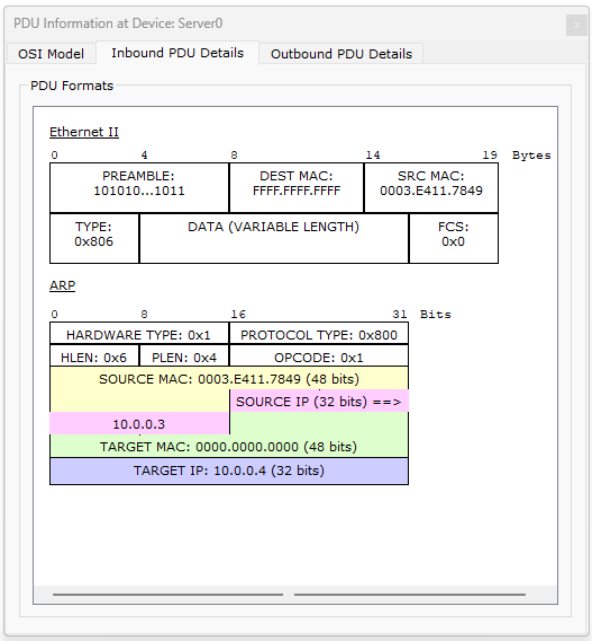


Figure 11.2: Inbound ARP, Server0

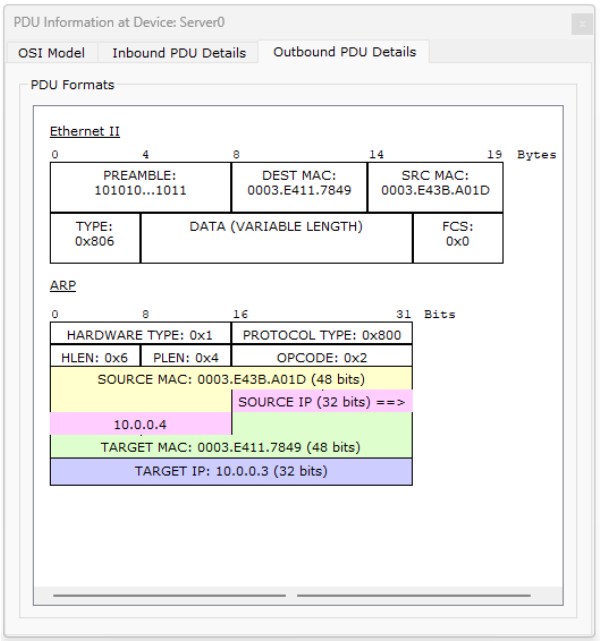


Figure 11.3: Outbound ARP, Server0

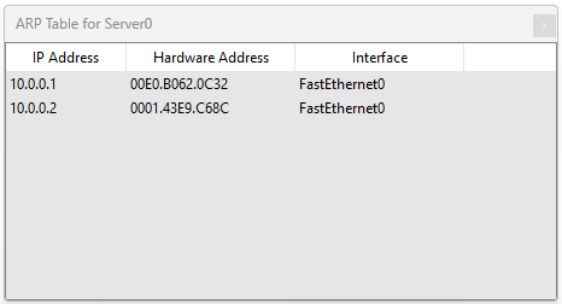


Figure 11.4: ARP Table, Server0

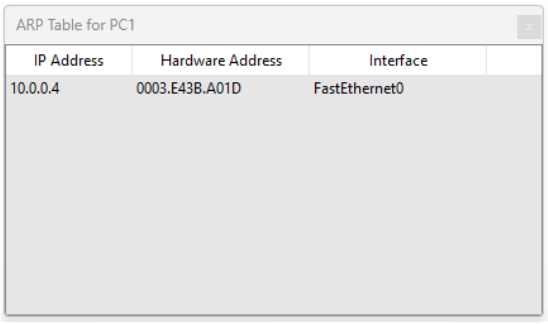
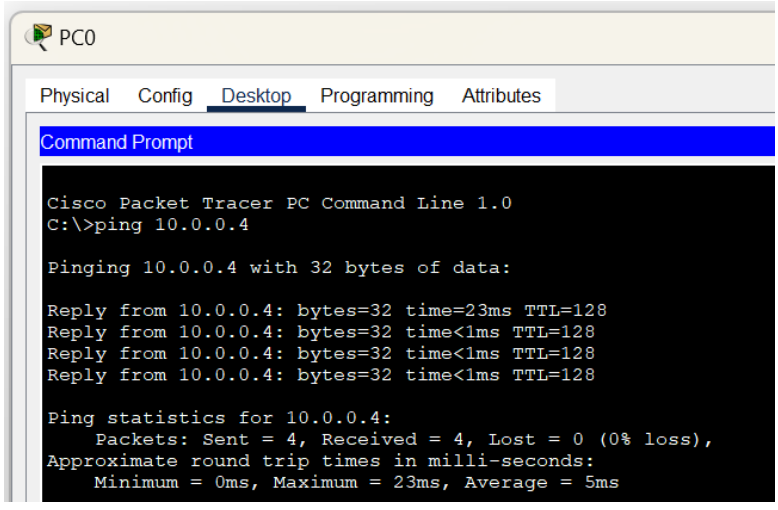


Figure 11.5: ARP Table, PC1

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Server0	ICMP		0.000	N	0	(edit)	



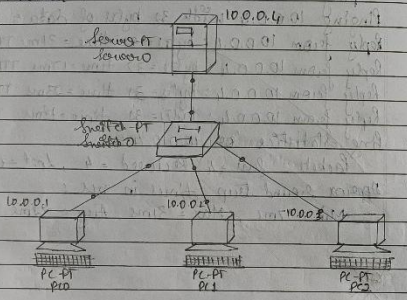
26.11.24

DATE: PAGE:

LABORATORY PROGRAM - 11

⇝ AIM OF THE EXPERIMENT : To demonstrate the working of address resolution protocol (ARP) within a LAN for communication.

⇝ TOPOLOGY :



⇝ CONFIGURATION :

- Connection : Use Copper Straight - Through for connection
- Now, Configuring end devices :
  - PC0 : IP address : 10.0.0.1
  - PC1 : IP address : 10.0.0.2
  - PC2 : IP address : 10.0.0.3
  - Server0 : IP address : 10.0.0.4 (In FE0/0)

Thus connections and configurations are done.

DATE: PAGE:

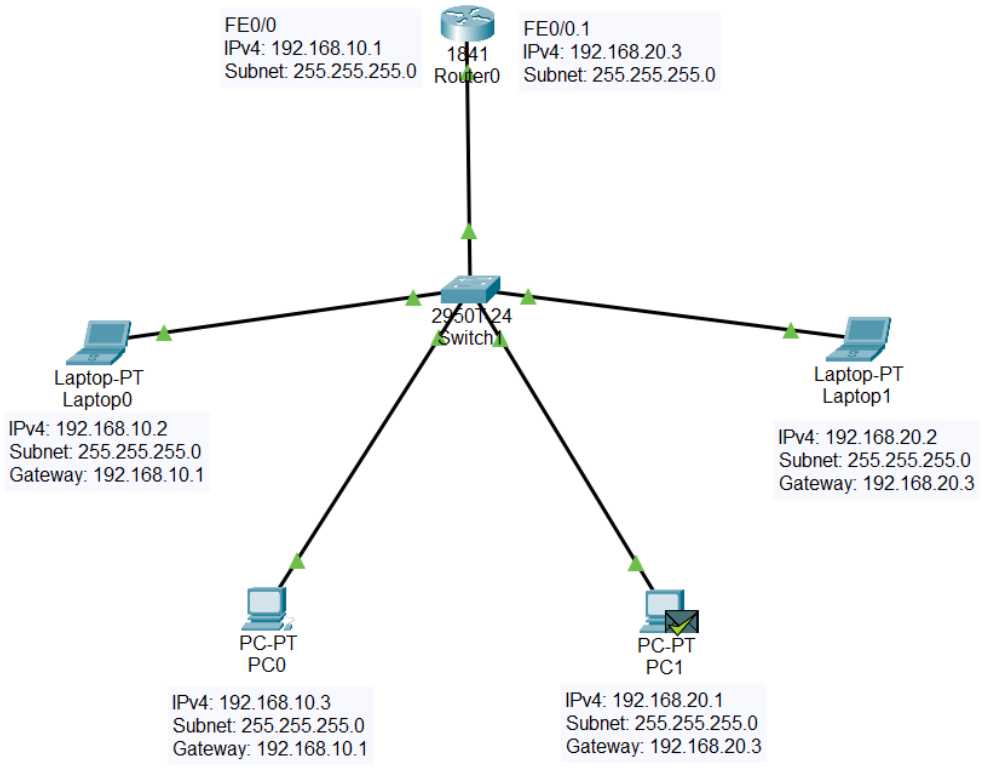
⇝ OBSERVATION :

- Initially, if we prompt :>ARP -a, the ARP table is empty.
- Now, lets PDU from PC1 to Server0, on clicking PDU, type ARP : we get ARP table describing :
  - SOURCE MAC : 0001.43E9.C68C (48 bits)
  - SOURCE IP : 10.0.0.2
  - TARGET MAC : 0000.0000.0000 (48 bits)
  - TARGET IP : 10.0.0.4 (32 bits)
- Now, on clicking PDU, type ARP of the Server0 : we get ARP table describing.
  - SOURCE MAC : 0003.E411.7849 (48 bits)
  - SOURCE IP : 10.0.0.4
  - TARGET MAC : 0003.E411.7849 (48 bits)
  - TARGET IP : 10.0.0.3 (32 bits)

Thus ARP table for each devices gets filled. Now communication is possible

LABORATORY PROGRAM – 12

To create a VLAN on top of the physical LAN and enable communication between physical LAN and virtual LAN.



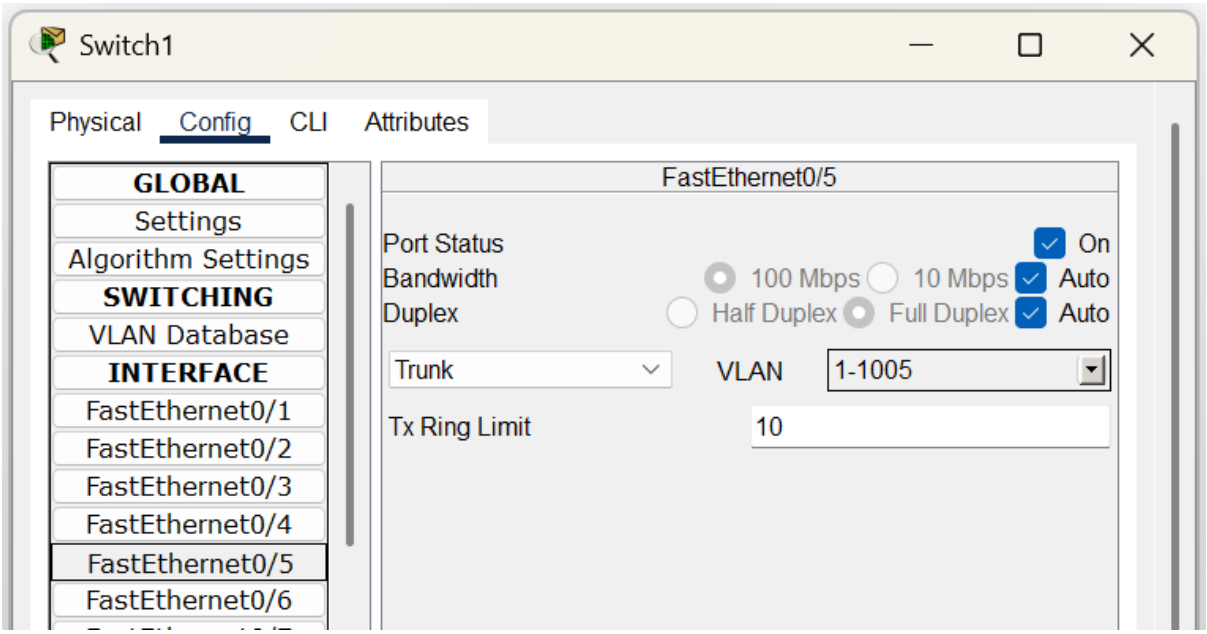


Figure 12.1: FE0/5 Switchport Trunk

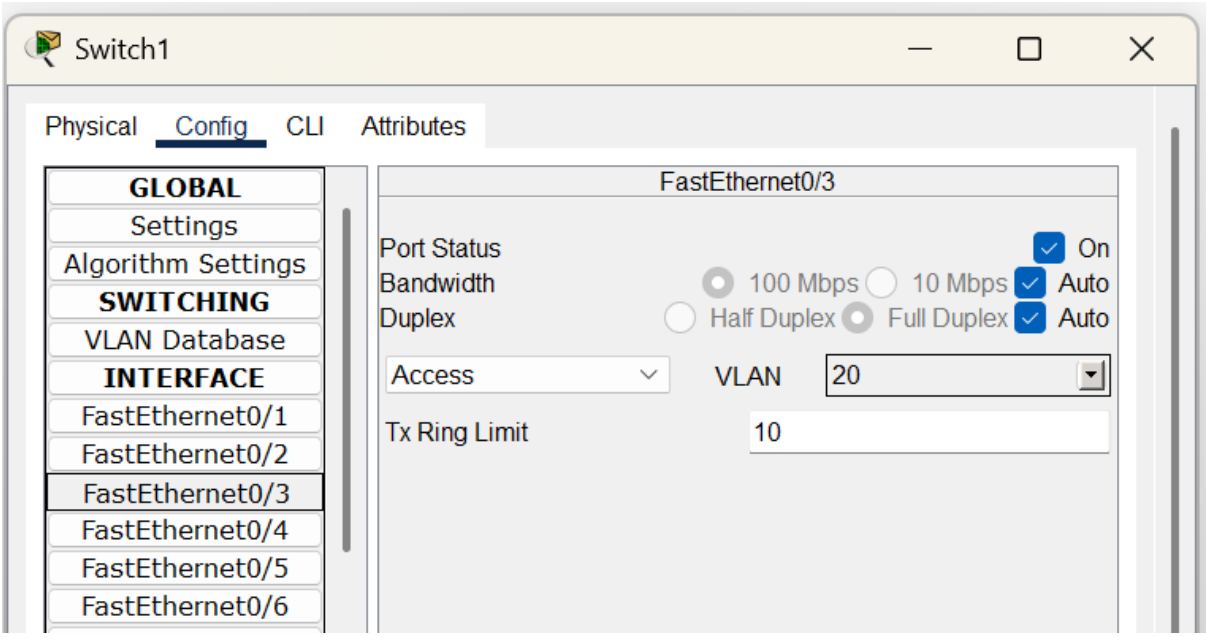


Figure 12.2: FE0/3 Switchport Access

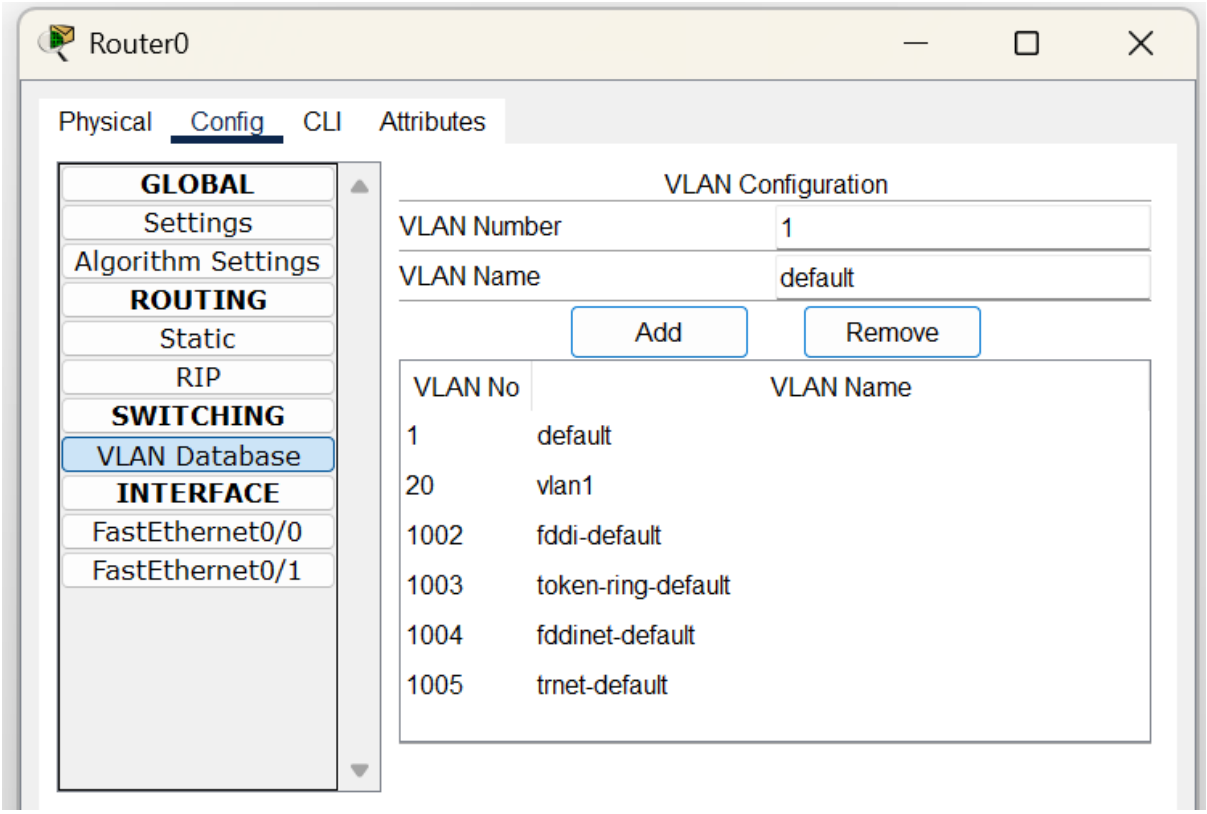


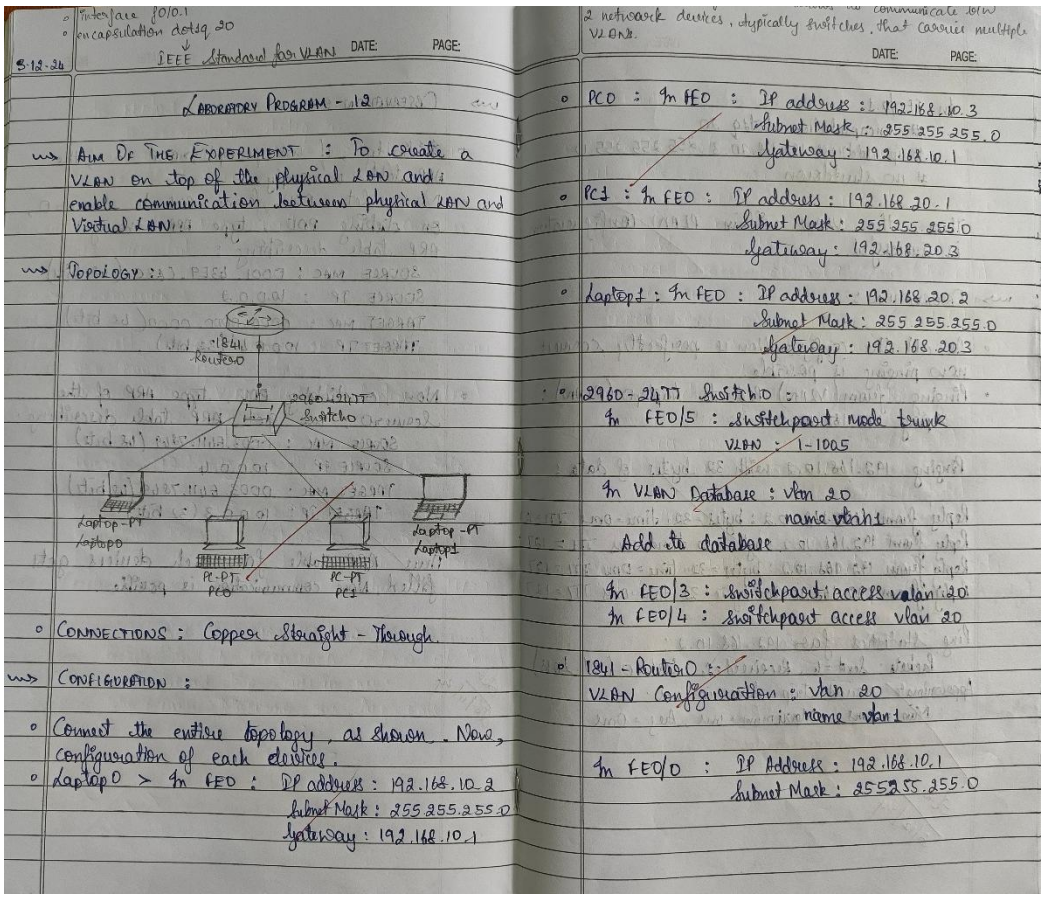
Figure 12.3: Router0 VLAN Database

```
Router(config)#interface FastEthernet0/0.1
Router(config-subif)#encapsulation dot1q 20
Router(config-subif)#ip address 192.168.20.3 255.255.255.0
Router(config-subif)#no shutdown
```

Figure 3: Router0, FE0/0.1

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC1	Router0	ICMP		0.000	N	0	(edit)	

```
C:\>ping 192.168.20.3

Pinging 192.168.20.3 with 32 bytes of data:

Reply from 192.168.20.3: bytes=32 time=2ms TTL=255
Reply from 192.168.20.3: bytes=32 time<1ms TTL=255
Reply from 192.168.20.3: bytes=32 time<1ms TTL=255
Reply from 192.168.20.3: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.20.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 0ms
```



DATE: PAGE:

In FED/0.1 :
encapsulation dot1q 20
ip address 192.168.20.3 255.255.255.0
no shutdown

Thus, the VLAN within PLAN configuration is successful.

==> OBSERVATION :

- As all the configuration is perfectly correct, now pinging is possible.
- Pinging from (VLAN) PC1 to Laptop0 (PLAN) :
  > ping 192.168.10.2

Pinging 192.168.10.2 with 32 bytes of data :

Reply from 192.168.10.2 : bytes = 32 time = 0ms TTL = 127
Reply from 192.168.10.2 : bytes = 32 time = 1ms TTL = 127
Reply from 192.168.10.2 : bytes = 32 time = 0ms TTL = 127
Reply from 192.168.10.2 : bytes = 32 time = 0ms TTL = 127

Ping statistics for 192.168.10.2 :
Packets : Sent = 4, Received = 4, Lost = 0 (0% Loss)
Approximate round trip times in ms :
Min = 0ms, Maximum = 1ms, Avg = 0ms.

3/12/24

LABORATORY PROGRAM – 13

Write a program for error detecting code using CRC-CCITT (8-bits).

Code

```
def xor(dividend, divisor):
    """Perform XOR operation between dividend and divisor."""
    result = ''
    for i in range(1, len(divisor)):
        result += '0' if dividend[i] == divisor[i] else '1'
    return result


def crc(data, gen_poly):
    """Compute the CRC check value using CRC-CCITT (8-bit)."""
    data_length = len(data)
    gen_length = len(gen_poly)

    # Append n-1 zeros to the data
    padded_data = data + '0' * (gen_length - 1)
    check_value = padded_data[:gen_length]

    for i in range(data_length):
        if check_value[0] == '1':
            # XOR operation if the first bit is 1
            check_value = xor(check_value, gen_poly)
        else:
            # Retain original check value if first bit is 0
            check_value = check_value[1:]
        # Shift left and add the next data bit
        if i + gen_length < len(padded_data):
            check_value += padded_data[i + gen_length]

    return check_value[1:]  # Remove the leading bit


def receiver(data, gen_poly):
    """Simulate the receiver side to check for errors."""
    print("\n-----------------------------")
    print("Data received:", data)

    # Perform CRC computation on received data
    remainder = crc(data, gen_poly)

    # Check if the remainder is all zeros
    if '1' in remainder:
        print("Error detected")
    else:
        print("No error detected")


if __name__ == "__main__":
    # Input data and generator polynomial
    data = input("Enter data to be transmitted: ")
    gen_poly = input("Enter the Generating polynomial: ")
```

```
    # Compute CRC check value
    check_value = crc(data, gen_poly)
    print("\n-----------------------------------")
    print("Data padded with n-1 zeros:", data + '0' * (len(gen_poly) - 1))
    print("CRC or Check value is:", check_value)

    # Append check value to data for transmission
    transmitted_data = data + check_value
    print("Final data to be sent:", transmitted_data)
    print("-----------------------------------\n")

    # Simulate the receiver side
    received_data = input("Enter the received data: ")
    receiver(received_data, gen_poly)
```

Output

```
Enter data to be transmitted: 1001100
Enter the Generating polynomial: 100001011

---------------------------------------
Data padded with n-1 zeros: 100110000000000
CRC or Check value is: 0100010
Final data to be sent: 10011000100010
---------------------------------------

Enter the received data: 10011000100011

-----------------------------
Data received: 10011000100011
Error detected
```

31.12.24 DATE: PAGE:

LABORATORY PROGRAM - 13

Write a program for error detecting code using CRC-CCITT (8-bits)

```
def xor(dividend, divisor):
    result = ''
    for i in range(1, len(divisor)):
        result += '0' if dividend[i] == divisor[i] else '1'
    return result

def crc(data, gen_poly):
    data_length = len(data)
    gen_length = len(gen_poly)

    padded_data = data + '0' * (gen_length - 1)
    check_value = padded_data[:gen_length]

    for i in range(data_length):
        if check_value[0] == '1':
            check_value = xor(check_value, gen_poly)
        else:
            check_value = check_value[1:]

        if i + gen_length < len(padded_data):
            check_value += padded_data[i + gen_length]
    return check_value[1:]
```

```
def receiver(data, gen_poly):
    print("\n----------")
    print("Data received:", data)

    remainder = crc(data, gen_poly)

    if '1' in remainder:
        print("Errors detected")
    else:
        print("No errors detected")

if __name__ == "__main__":
    data = input("Enter data to be transmitted:")
    gen_poly = input("Enter the generating polynomial")

    check_value = crc(data, gen_poly)
    print("\n----------")
    print("Data padded with n-1 zeros:", data + '0'*(len(gen_poly)-1))
    print("CRC or Check value is:", check_value)

    transmitted_data = data + check_value
    print("Final data to be sent:", transmitted_data)
    print("----------\n")
received_data = input("Enter the received data:")
receiver(received_data, gen_poly)
```

OUTPUT :

```
Enter data to be transmitted: 1001100
Enter the generating polynomial: 1000010011

Data padded with n-1 zeros : 100110000000000
CRC or Check value is : 0100010
Final data to be sent : 1001100010001010

Enter the received data : 1001100010001011

Data received : 1001100010001011
Errors detected
```

How does the CRC obtained: The data to sent is adjuncted with n-1 zeroes where n is the length of key then binary division of the adjunct data by key is done the remainder is the checksum code, that is XORed with adjuncted data.

LABORATORY PROGRAM – 14

Write a program for congestion control using Leaky bucket algorithm.

Code

```
# Getting user inputs
storage = int(input("Enter initial packets in the bucket: "))
no_of_queries = int(input("Enter total no. of times bucket content is checked: "))
bucket_size = int(input("Enter total no. of packets that can be accommodated in the bucket:
"))
input_pkt_size = int(input("Enter no. of packets that enters the bucket at a time: "))
output_pkt_size = int(input("Enter no. of packets that exits the bucket at a time: "))

for i in range(no_of_queries):  # space left
    size_left = bucket_size - storage
    if input_pkt_size <= size_left:
        # update storage
        storage += input_pkt_size
    else:
        print("Packet loss =", input_pkt_size)

    print(f"Buffer size = {storage} out of bucket size = {bucket_size}")

    # as packets are sent out into the network, the size of the storage decreases
    storage -= output_pkt_size
```

Output

```
Enter initial packets in the bucket: 0
Enter total no. of times bucket content is checked: 4
Enter total no. of packets that can be accommodated in the bucket: 10
Enter no. of packets that enters the bucket at a time: 4
Enter no. of packets that exits the bucket at a time: 1
Buffer size = 4 out of bucket size = 10
Buffer size = 7 out of bucket size = 10
Buffer size = 10 out of bucket size = 10
Packet loss = 4
Buffer size = 9 out of bucket size = 10
```

17.12.24

DATE: PAGE:

LABORATORY PROGRAM - 14

Write a program for congestion control using leaky bucket algorithm.

```
storage = int(input("Enter initial packets in the buckets: "))
no of queries = int(input("Enter total no of times bucket content is checked: "))
bucket size = int(input("Enter total no of packets that can be accommodated in the bucket: "))
input pkt size = int(input("Enter no of packets that enters the bucket at a time: "))
output pkt size = int(input("Enter no of packets that exits the bucket at a time: "))

for i in range(no of queries):
    size left = bucket size - storage
    if input pkt size <= size left:
        storage += input pkt size
    else:
        print("Packet loss = ", input pkt size)

    print(f"Buffer size = {storage} out of bucket size = {bucket size}")

    storage -= output pkt size
```

DATE: PAGE:

OUTPUT:

```
Enter initial packets in the bucket: 0
Enter total no. of times bucket content is checked: 4
Enter total no. of packets that can be accomodate in the bucket: 10
Enter no. of packets that enters the bucket at a time: 4
Enter no of packets that exits the bucket at a time: 1
Buffer size = 4 out of bucket size = 10
Buffer size = 7 out of bucket size = 10
Buffer size = 10 out of bucket size = 10
Packet loss = 4
Buffer size = 9 out of bucket size = 10
```

17/12/24

LABORATORY PROGRAM – 15(A)

Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Code: Client.py

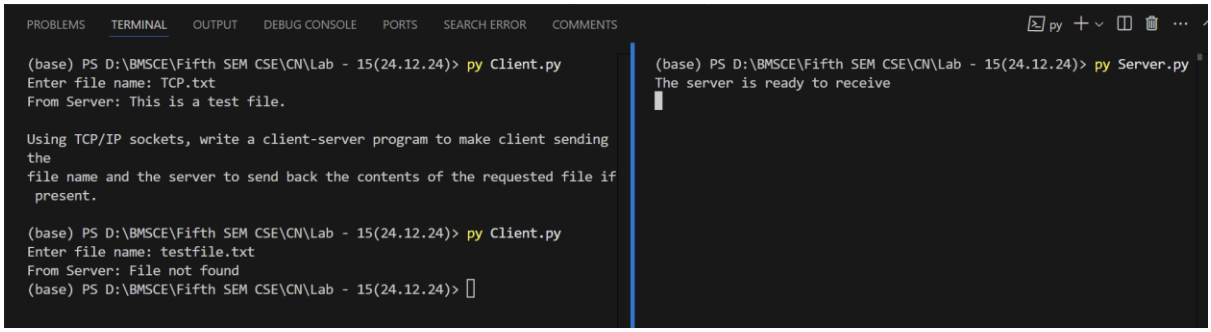
```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number where the server listens

# Create TCP socket
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))  # Connect to server

# Ask user for file name to request
sentence = input("Enter file name: ")

# Send file name to server
clientSocket.send(sentence.encode())

# Receive file contents from server
filecontents = clientSocket.recv(1024).decode()
print('From Server:', filecontents)

# Close the connection
clientSocket.close()
```

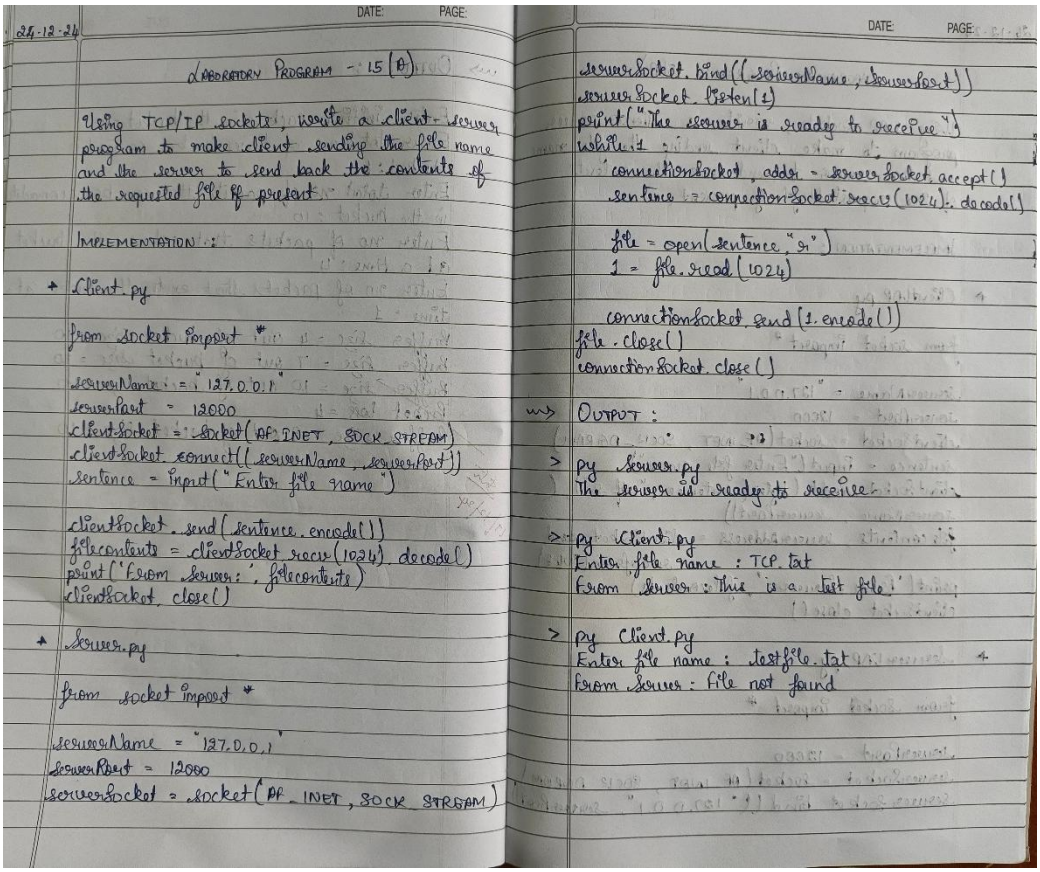
Code: Server.py

```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number to listen on

# Create TCP socket
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))  # Bind socket to the address and port
serverSocket.listen(1)  # Listen for 1 connection
print("The server is ready to receive")

while True:
    # Accept a connection
    connectionSocket, addr = serverSocket.accept()

    # Receive the file name from the client
    sentence = connectionSocket.recv(1024).decode()

    # Try opening the file
    try:
        file = open(sentence, "r")  # Open file in read mode
        fileContents = file.read(1024)  # Read file content (up to 1024 bytes)
```

```
    connectionSocket.send(fileContents.encode())  # Send file contents to client
    file.close()
except FileNotFoundError:
    # Send error message if file not found
    connectionSocket.send("File not found".encode())

# Close the connection
connectionSocket.close()
```

Output





LABORATORY PROGRAM – 15(B)

Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Code: ClientUDP.py

```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number where the server listens

# Create UDP socket
clientSocket = socket(AF_INET, SOCK_DGRAM)

# Ask user for file name to request
sentence = input("Enter file name: ")

# Send the file name to the server using UDP
clientSocket.sendto(sentence.encode("utf-8"), (serverName, serverPort))

# Receive file contents from the server
fileContents, serverAddress = clientSocket.recvfrom(2048)

# Print the file contents received from the server
print("From Server:", fileContents.decode())

# Close the UDP socket
clientSocket.close()
```

Code: ServerUDP.py

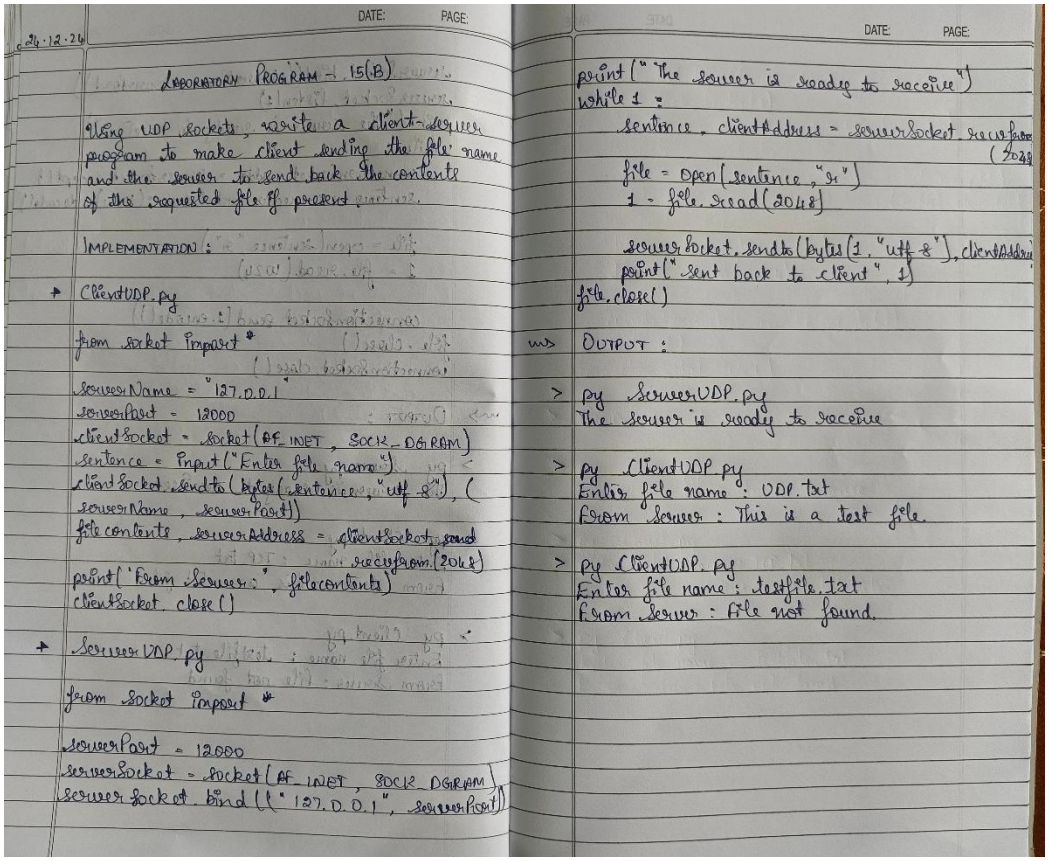
```
from socket import *
serverPort = 12000  # Port number to listen on

# Create UDP socket
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))  # Bind the socket to the server address and port

print("The server is ready to receive")

while True:
    # Receive file name from the client
    sentence, clientAddress = serverSocket.recvfrom(2048)

    # Try opening the file
    try:
        file = open(sentence.decode(), "r")  # Open file in read mode
        fileContents = file.read(2048)  # Read file content (up to 2048 bytes)
        serverSocket.sendto(fileContents.encode("utf-8"), clientAddress)  # Send file contents to
client
        file.close()
    except FileNotFoundError:
```

```
# Send error message if file not found
serverSocket.sendto("File not found".encode("utf-8"), clientAddress)
```

Output

```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE  PORTS  SEARCH ERROR  COMMENTS                    pwsh

(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ClientUDP.py
Enter file name: UDP.txt
From Server: This is a test file.

Using UDP sockets, write a client-server program to make client sending
the file
name and the server to send back the contents of the requested file if p
resent.
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ClientUDP.py
Enter file name: testfile.txt
From Server: File not found
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)>
```

```
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ServerUDP.py
The server is ready to receive
```



24.12.24

LABORATORY PROGRAM - 15(B)

Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

IMPLEMENTATION :

ClientUDP.py

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("Enter file name:")
clientSocket.sendto(bytes(sentence,"utf-8"),(
serverName, serverPort))
fileContents, serverAddress = clientSocket.recvfrom(2048)
print('From Server:', fileContents)
clientSocket.close()
```

ServerUDP.py

```
from socket import *

serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to receive")
while 1 :
    sentence, clientAddress = serverSocket.recvfrom
                                        (2048)
    file = open(sentence, "r")
    l = file.read(2048)

    serverSocket.sendto(bytes(l,"utf-8"),clientAddress)
    print("sent back to client", l)
file.close()
```

OUTPUT :

```
> py ServerUDP.py
The server is ready to receive

> py ClientUDP.py
Enter file name: UDP.txt
From Server : This is a test file.

> py ClientUDP.py
Enter file name : testfile.txt
From Server : File not found.
```